

Министерство науки и высшего образования Российской Федерации  
Федеральное государственное автономное образовательное учреждение  
высшего образования  
«САНКТ-ПЕТЕРБУРГСКИЙ УНИВЕРСИТЕТ  
АЭРОКОСМИЧЕСКОГО ПРИБОРОСТРОЕНИЯ»

КАФЕДРА № 51

ОТЧЕТ  
ЗАЩИЩЕН С ОЦЕНКОЙ  
ПРЕПОДАВАТЕЛЬ

Д.Т.Н., доцент  
\_\_\_\_\_  
должность, уч. степень, звание

\_\_\_\_\_  
подпись, дата

Мошак Н. Н.  
\_\_\_\_\_  
инициалы, фамилия

ОТЧЕТ О ЛАБОРАТОРНОЙ РАБОТЕ №3

«АДМИНИСТРИРОВАНИЕ И НАСТРОЙКА ПОЛИТИКИ БЕЗОПАСНОСТИ  
СЕРВЕРА РЕЛЯЦИОННОЙ БАЗЫ ДАННЫХ»

по курсу: Безопасность информационных систем

РАБОТУ ВЫПОЛНИЛ

СТУДЕНТ ГР. № 5712

\_\_\_\_\_  
подпись, дата

Е. В. Митрофанова  
\_\_\_\_\_  
инициалы, фамилия

Санкт-Петербург 2020

## **1. Цель работы**

Изучить команды MySQL и систему привилегий (privilege system). Научиться устанавливать и администрировать SQL-сервер на примере сервера MySQL, а также настраивать его параметры безопасности.

Используемое программное обеспечение: ОС версии не ниже Windows 7, ПО сервера MySQL.

## **2. Краткие теоретические сведения**

### **2.1. Реляционные базы данных. Общие сведения**

Задача длительного хранения и обработки информации появилась практически сразу с появлением первых компьютеров. Для решения этой задачи в конце 60-х годов были разработаны специализированные программы, получившие название систем управления базами данных (СУБД). СУБД проделали длительный путь эволюции от системы управления файлами, через иерархические и сетевые базы данных. В конце 80-х годов доминирующей стала система управления реляционными базами данных (СУРБД). С этого времени такие СУБД стали стандартом де-факто, и для того, чтобы унифицировать работу с ними, был разработан структурированный язык запросов (SQL), который представляет собой язык управления именно реляционными базами данных.

Существуют следующие разновидности баз данных:

- иерархические;
- реляционные;
- объектно-ориентированные;
- гибридные.

**Иерархическая** база данных основана на древовидной структуре хранения информации. В этом смысле иерархические базы данных очень напоминают файловую систему компьютера.

В **реляционных** базах данных данные собраны в таблицы, которые в свою очередь состоят из столбцов и строк, на пересечении которых расположены ячейки. Запросы к таким базам данных возвращает таблицу, которая повторно может участвовать в следующем запросе. Данные в одних таблицах, как правило, связаны с данными других таблиц, откуда и произошло название «реляционные».

В **объектно-ориентированных** базах данных данные хранятся в виде объектов. С объектно-ориентированными базами данных удобно работать, применяя объектно-ориентированное программирование. Однако, на сегодняшний день такие базы данных еще не достигли популярности реляционных, поскольку пока значительно уступают им в производительности.

**Гибридные** СУБД совмещают в себе возможности реляционных и объектно-ориентированных баз данных. Эти модели характеризуются простотой структуры данных, удобным для пользователя табличным представлением и возможностью использования формального аппарата алгебры отношений и реляционного исчисления для обработки данных.

Понятие реляционный (англ. *relation* — отношение) связано с разработками известного английского специалиста в области систем баз данных Эдгара Кодда (Edgar Codd). Модель реляционной базы данных представляет данные в виде таблиц, разбитых на строки и столбцы, на пересечении которых находятся данные. Кратко особенности реляционной базы данных можно описать следующим образом:

- Данные хранятся в таблицах, состоящих из столбцов и строк;
- На пересечении каждого столбца и строчки стоит в точности одно значение;
- У каждого столбца есть своё имя, которое служит его названием, и все значения в одном столбце имеют один тип. Например, в столбце `id_forum` все значения имеют целочисленный тип, а в строке `name` - текстовый;
- Столбцы располагаются в определённом порядке, который определяется при создании таблицы, в отличие от строк, которые располагаются в произвольном порядке. В таблице может не быть не одной строчки, но обязательно должен быть хотя бы один столбец;
- Запросы к базе данных возвращают результат в виде таблиц, которые тоже могут выступать как объект запросов.

Для работы с базами данных используется язык SQL (Structured Query Language — язык структурированных запросов). Стандарт SQL определен ANSI (American National Standard Institute). SQL предназначен для манипуляции данными, которые хранятся в Системах управления реляционными базами данных (RDBMS). SQL имеет команды, с помощью которых данные можно извлекать, сортировать, обновлять, удалять и добавлять. Стандарты языка SQL определяет ANSI (American National Standards Institute). Однако SQL не является изобретением ANSI, он – продукт исследования фирмы IBM, проводимого в начале 70-х годов 20 века. Другие организации и учебные заведения также внесли вклад в создание этого языка, например, компания Oracle или Калифорнийский университет Беркли. В настоящее время действует стандарт, принятый в 2003 году (SQL-3).

SQL можно использовать с такими базами RDBMS как MySQL, mSQL, PostgreSQL, Oracle, Microsoft SQL Server, Access, Sybase, Ingres. Эти системы RDBMS поддерживают все важные и общепринятые операторы SQL, однако каждая из них имеет множество своих собственных патентованных операторов и расширений.

SQL является общим языком запросов для нескольких баз данных различных типов.

## **2.2. База данных MySQL. Общие сведения**

MySQL, которая является RDBMS с открытым исходным кодом, доступна для загрузки на сайте MySQL.com. Разработчиком MySQL является компания MySQL AB. В настоящее время компания куплена корпорацией Oracle, которой и принадлежит теперь продукт. Свое происхождение MySQL ведет от продукта mSQL, разработанного в конце 1970-х гг. компанией TcX и использовавшемуся для доступа к таблицам, для которых использовались

собственные быстрые подпрограммы низкого уровня. Однако после тестирования был сделан вывод, что скорость и гибкость mSQL недостаточны. В результате для базы данных был разработан новый SQL-интерфейс. Новый продукт получил название MySQL.

Вот как характеризуют MySQL её разработчики.

- MySQL - это система управления базами данных.

База данных представляет собой структурированную совокупность данных. Эти данные могут быть любыми - от простого списка предстоящих покупок до перечня экспонатов картинной галереи или огромного количества информации в корпоративной сети. Для записи, выборки и обработки данных, хранящихся в компьютерной базе данных, необходима система управления базой данных, каковой и является ПО MySQL. Поскольку компьютеры замечательно справляются с обработкой больших объемов данных, управление базами данных играет центральную роль в вычислениях. Реализовано такое управление может быть по-разному - как в виде отдельных утилит, так и в виде кода, входящего в состав других приложений.

- MySQL - это система управления реляционными базами данных.

В реляционной базе данные хранятся в отдельных таблицах, благодаря чему достигается выигрыш в скорости и гибкости. Таблицы связываются между собой при помощи отношений, благодаря чему обеспечивается возможность объединять при выполнении запроса данные из нескольких таблиц. SQL как часть системы MySQL можно охарактеризовать как язык структурированных запросов плюс наиболее распространенный стандартный язык, используемый для доступа к базам данных.

- Программное обеспечение MySQL - это ПО с открытым кодом.

ПО с открытым кодом означает, что применять и модифицировать его может любой желающий. Такое ПО можно получать по Internet и использовать бесплатно. При этом каждый пользователь может изучить исходный код и изменить его в соответствии со своими потребностями. Укажем важные характеристики программного обеспечения MySQL:

- Внутренние характеристики и переносимость
  - Написан на C и C++. Протестирован на множестве различных компиляторов.
  - Работает на различных аппаратных платформах и разных операционных системах.
  - Высокая производительность за счет максимально оптимизированного кода, эффективной системы распределения памяти и продуманной системы дисковых таблиц.
- Масштабируемость
  - Способность работать с очень большими базами данных (десятки и сотни миллионов записей).
  - Возможность кластеризации серверов и распределения обработки информации между серверами
- Технические возможности СУБД MySQL

ПО MySQL является системой клиент-сервер, которая содержит многопоточный SQL-сервер, обеспечивающий поддержку различных вычислительных машин баз данных, а также несколько различных клиентских программ и библиотек, средства администрирования и широкий спектр программных интерфейсов (API).

- **Безопасность**

Система безопасности основана на привилегиях и паролях с возможностью верификации с удаленного компьютера, за счет чего обеспечивается гибкость и безопасность. Пароли при передаче по сети при соединении с сервером шифруются. Клиенты могут соединяться с MySQL, используя сокет TCP/IP, сокет Unix или именованные каналы (named pipes, под NT).

Клиентская программа MySQL представляет собой утилиту командной строки. Эта программа подключается к серверу по сети. Команды, выполняемые сервером, обычно связаны с чтением и записью данных на жестком диске. Клиентские программы могут работать не только в режиме командной строки. Есть и графические клиенты, например MySQL GUI, [phpMyAdmin](#) и др.

### 2.3. **Краткий обзор команд MySQL**

Создание базы данных выполняется с помощью команды **CREATE DATABASE**.

Синтаксис команды:

**CREATE DATABASE** *database\_name*

*database\_name* - Имя, которое будет присвоено создаваемой базе данных.

Для **удаления** базы данных используется команда **DROP DATABASE**.

Синтаксис:

**DROP DATABASE** *database\_name*

где

*database\_name* - задает имя базы данных, которую необходимо удалить.

**Создание** таблицы производится командой **CREATE TABLE**.

**CREATE TABLE** *table\_name*(*column\_name1 type*, *column\_name2 type*,...)

*table\_name* - имя новой таблицы;

*column\_name* - имена колонок (полей), которые будут присутствовать в создаваемой таблице.

*type* - определяет тип данных создаваемой колонки.

Например, надо создать таблицу учетных записей и паролей с именем *user\_pass*. Пусть таблица будет состоять из двух столбцов – *UserName* и *UserPassword* с типом данных *text*:

**CREATE TABLE** *user\_pass*(*UserName text*, *UserPassword text*);

**Удаление** таблицы производится командой **DROP TABLE**

**DROP TABLE** *table\_name*

*table\_name* - имя удаляемой таблицы.

**Вставка записи** осуществляется командой **INSERT INTO**

```
INSERT INTO table_name(field_name1, field_name2,...) values('content1',  
'content2',...)
```

Данная команда добавляет в таблицу *table\_name* запись, у которой поля, обозначенные как *field\_nameN*, установлены в значение *contentN*.

Например, в таблицу учетных записей и паролей можно добавить запись следующим образом:

```
INSERT INTO user_pass(UserName, UserPassword)  
values('Operator', '1qAz_weR');
```

**Поиск записей** осуществляется командой **SELECT**

```
SELECT * FROM table_name WHERE (выражение) [order by field_name  
[desc][asc]]
```

Эта команда ищет все записи в таблице *table\_name*, которые удовлетворяют выражению *выражение*.

Если записей несколько, то при указанном предложении *order by* они будут отсортированы по тому полю, имя которого записывается правее этого ключевого слова (если задано слово *desc*, то упорядочивание происходит в обратном порядке). В предложении *order by* могут также задаваться несколько полей. Особое значение имеет символ *\**. Он предписывает, что из отобранных записей следует извлечь все поля, когда будет выполнена команда получения выборки. С другой стороны, вместо звездочки можно через запятую непосредственно перечислить имена полей, которые требуют извлечения.

Например, выбрать все записи из таблицы можно следующим образом:

```
SELECT * FROM user_pass;
```

Выбрать все записи с сортировкой по имени в обратном алфавитном порядке:

```
SELECT * FROM user_pass ORDER BY UserName desc;
```

Выбрать пароль для пользователя Admin:

```
SELECT UserPassword FROM user_pass WHERE UserName='Admin';
```

### **3. Ход работы.**

1) Установка операционной системы Windows 7 на диск C.

2) Установить MySQL на платформу.

Дистрибутив был скачан с сайта [www.mysql.com](http://www.mysql.com). Для установки необходимо иметь права администратора.

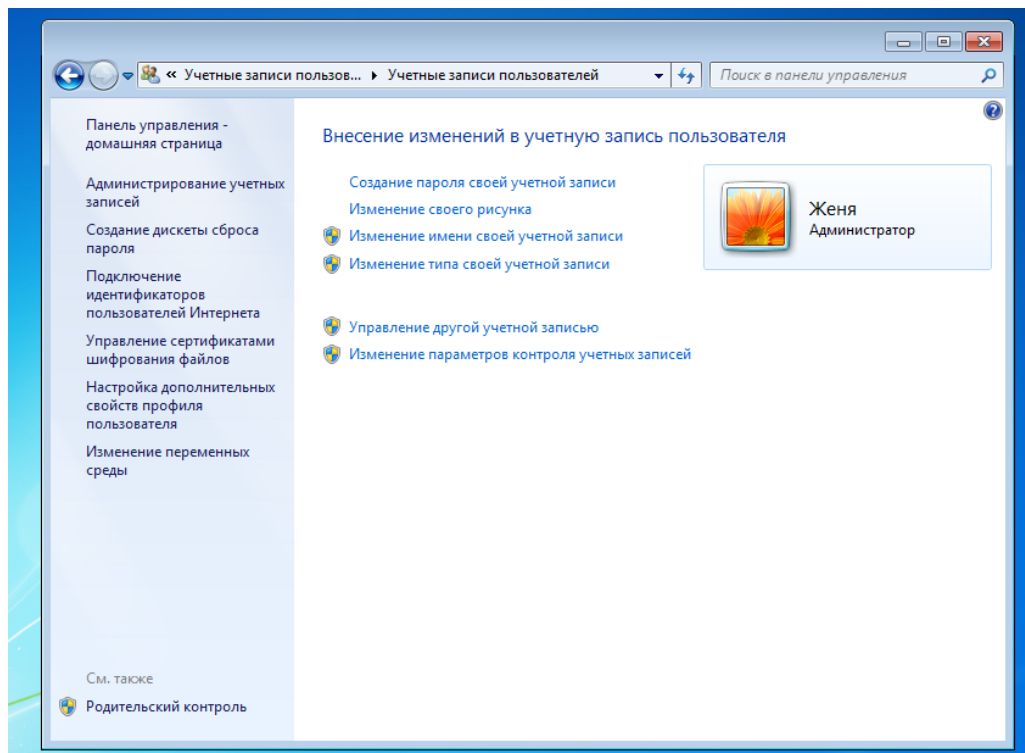


Рисунок 1 – Подтверждение прав администратора

Обязательно должны быть установлены:

- В группе «MySQL Server 5.6.11» - MySQL Server, Client Programs, Server Data Files.
- В группе Applications – MySQL Notifier.
- Группу MySQL Connectors – не устанавливать
- Группу Documentation – рекомендуется установить полностью.

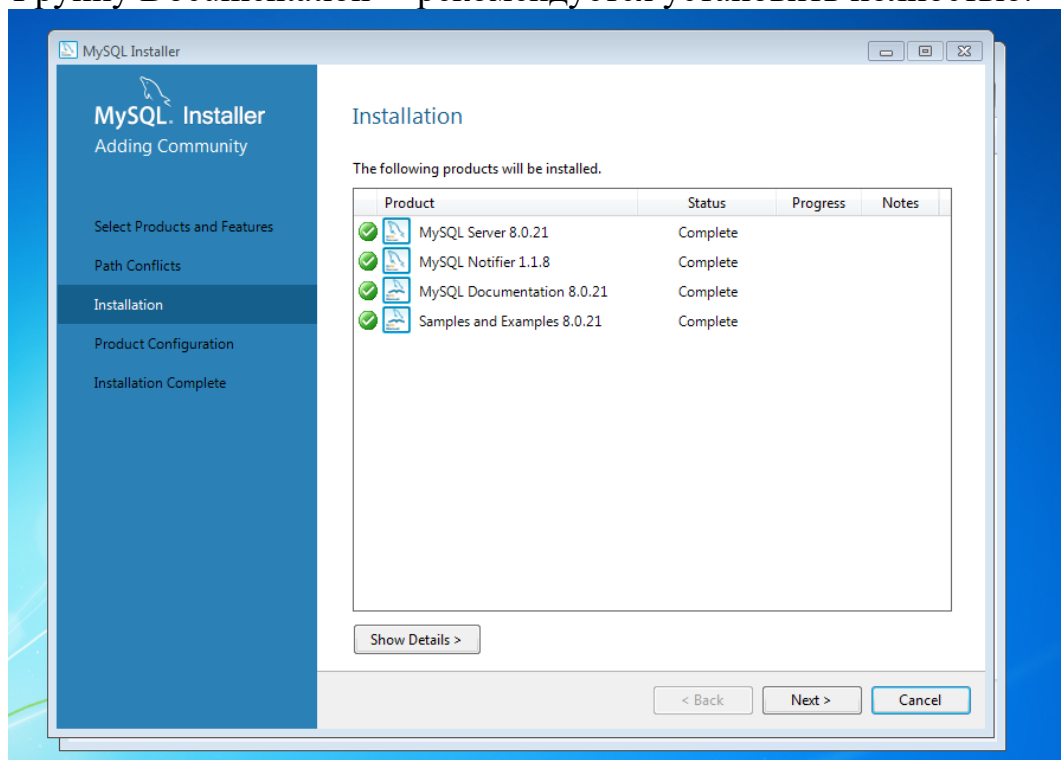


Рисунок 2 – Выбор компонента для установки

Окно настройки сетевой части. Необходимо выбрать конфигурацию «Development Machine» - этот тип установки предназначен для разработки и тестирования сайтов, в этом случае ресурсы компьютера будут подвергаться минимальной нагрузке.

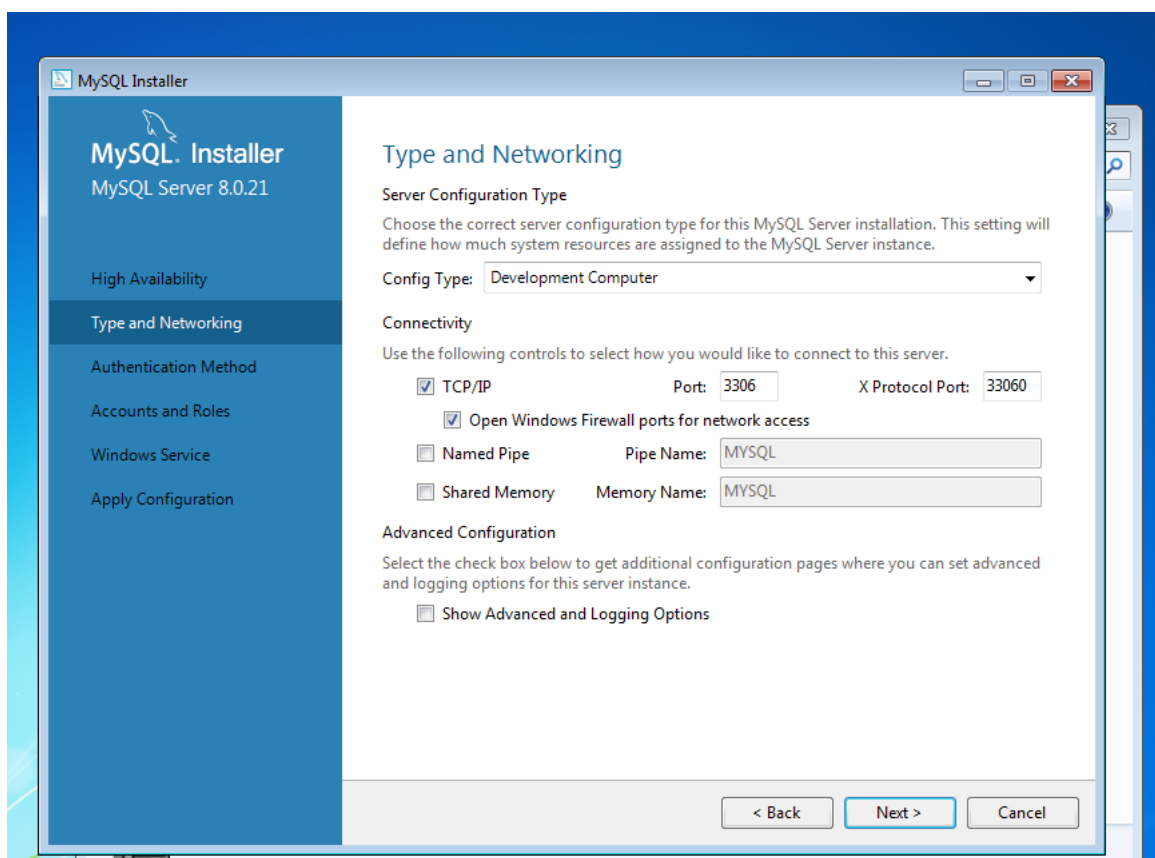


Рисунок 3 – Окно настройки серверной части

В следующем окне необходимо настроить пароль главного администратора сервера. Оставлять это поле пустым не рекомендуется, это негативно скажется на безопасности.

Т.к. в требованиях к типу ИС 1Б указано, что «должны осуществляться идентификация и проверка подлинности субъектов доступа при входе в систему по идентификатору (коду) и паролю временного действия длиной не менее восьми буквенно-цифровых символов», то необходимо установить пароль состоящий не менее чем из 8 символов.

### Accounts and Roles

#### Root Account Password

Enter the password for the root account. Please remember to store this password in a secure place.

MySQL Root Password:

Repeat Password:

Password Strength: **Strong**

Рисунок 4 - Настройка пароля



Программа установлена успешно, проверки пройдены.

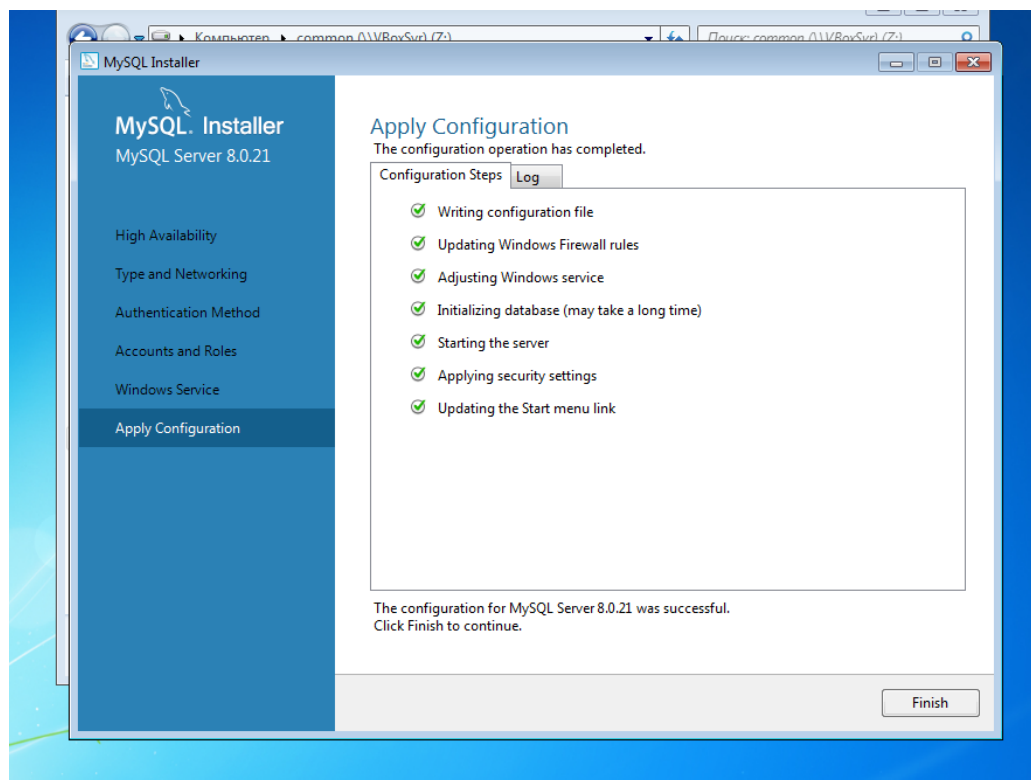


Рисунок 5 - Положительная проверка

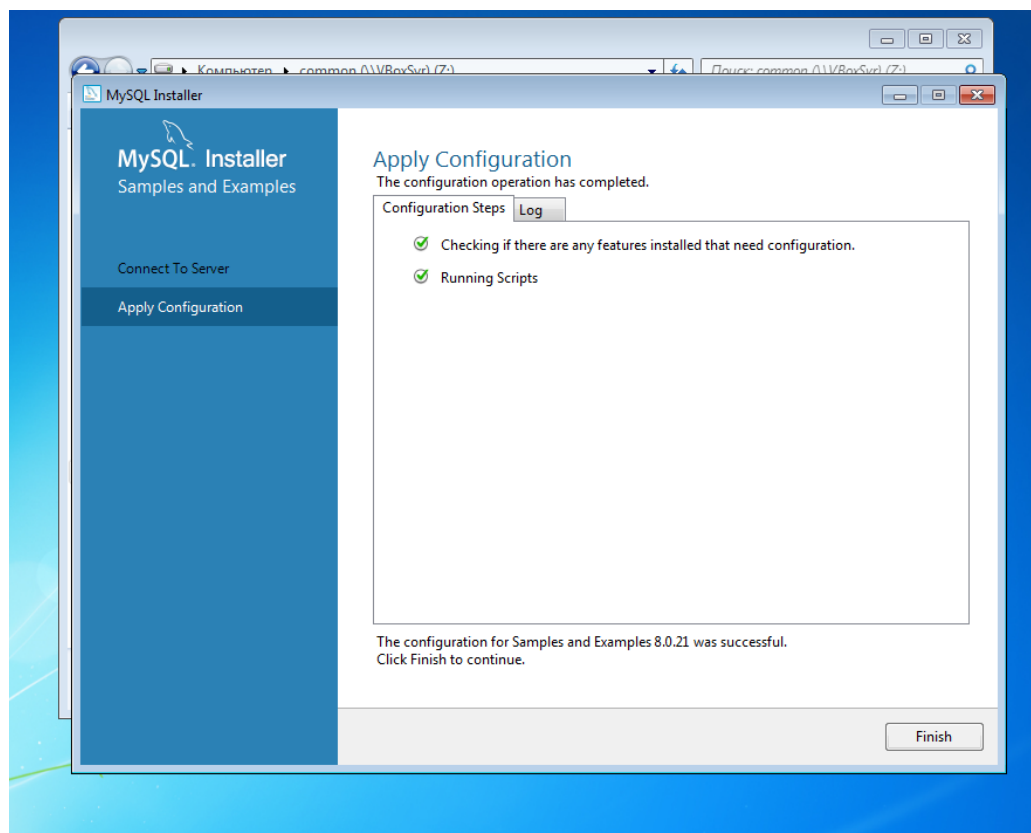


Рисунок 6 - Положительная проверка

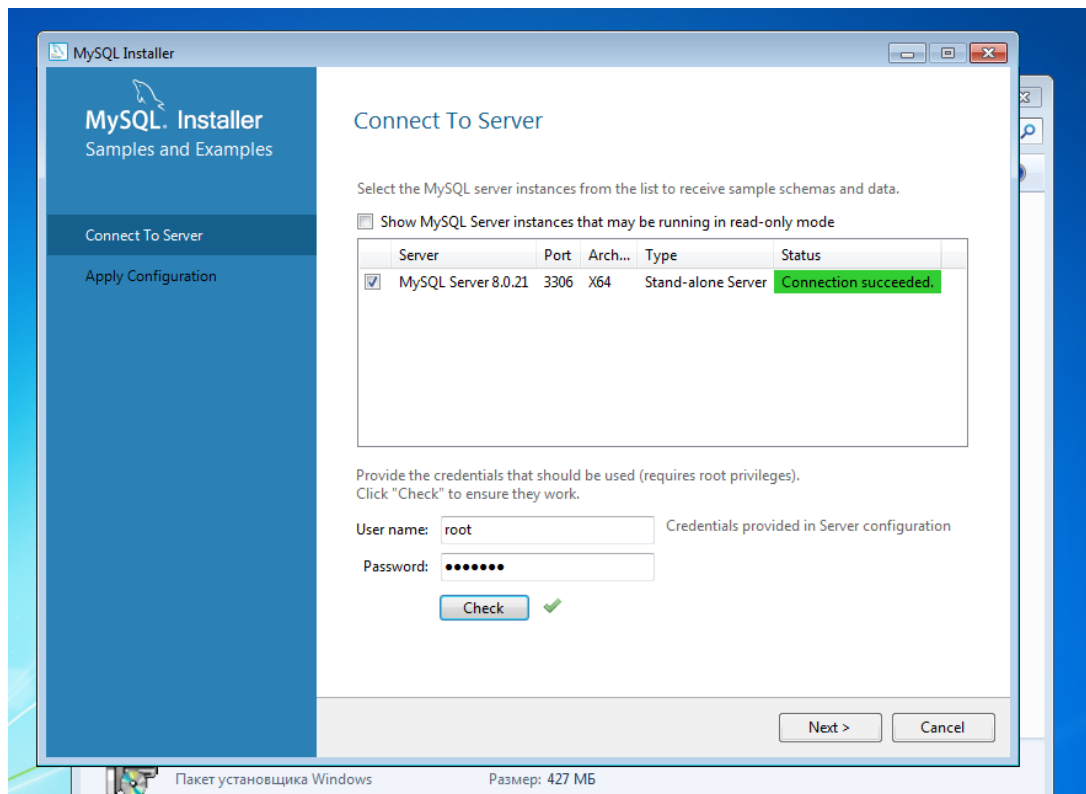


Рисунок 7 - Положительная проверка

Для управления сервером используется утилита «MySQL Notifier» (вызывается из меню программ). Утилита выводит иконку в панели задач, являющуюся индикатором состояния сервера базы данных, а также позволяющую управлять сервером (запуск и останов сервера).

### 3) Создание базы данных MySQL.

На данный момент имеется настроенный сервер «MySQL Server» и утилита для взаимодействия с ним «MySQL Notifier». Для того, чтобы можно было использовать возможности сервера, такие как создание и назначение прав пользователям на таблицы и базы данных, требуется, собственно, сначала создать пользовательскую базу данных и таблицу в ней. Имеющиеся по умолчанию базы данных можно посмотреть с использованием команды `show databases;`

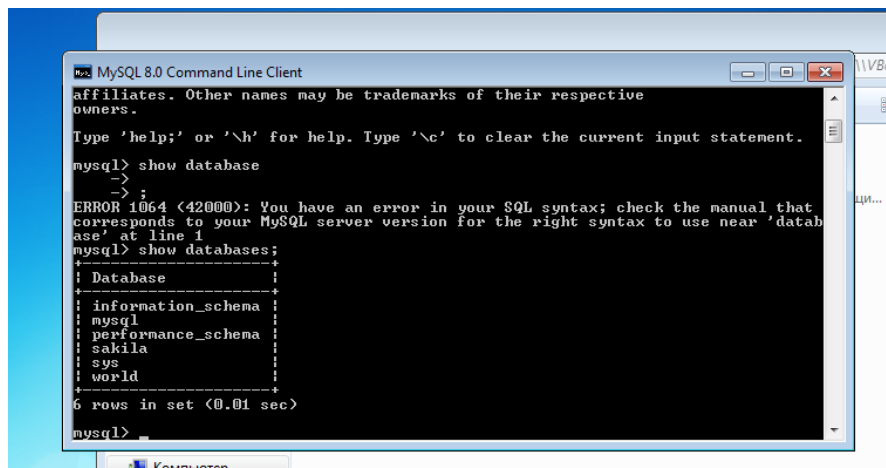
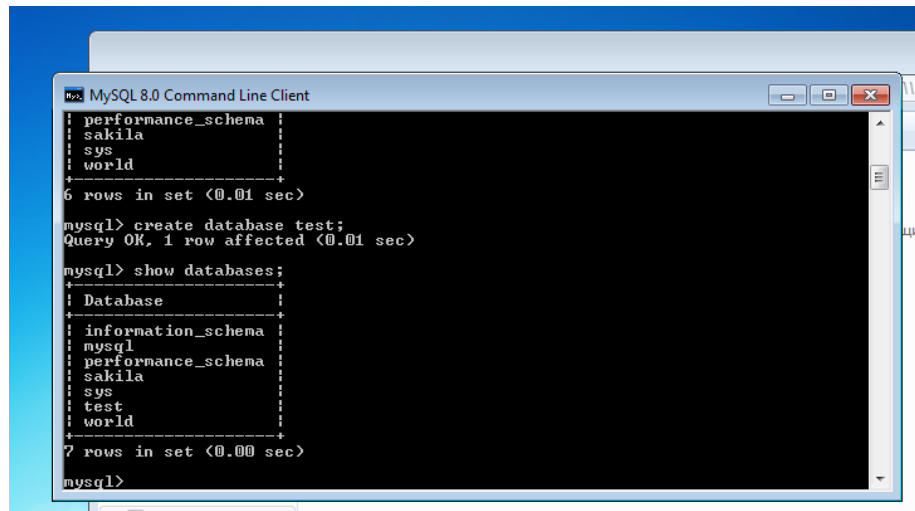


Рисунок 8 – Имеющиеся по умолчанию базы данных

Поскольку среди перечисленных баз данных имеются те, изменение данных в которых нежелательно (в частности, а базе данных mysql находится таблица user с перечнем всех пользователей, их паролей и способов подключения к базам данных), создадим собственную базу данных. с помощью следующей команды create database test. Проверим корректность создания.



```
mysql> show databases;
+-----+
| Database |
+-----+
| information_schema |
| mysql      |
| performance_schema |
| sakila     |
| sys       |
| test      |
| world     |
+-----+
7 rows in set (0.00 sec)

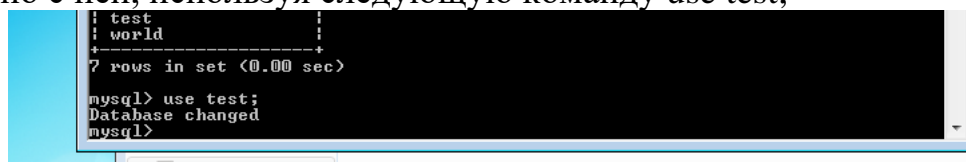
mysql> create database test;
Query OK, 1 row affected (0.01 sec)

mysql> show databases;
+-----+
| Database |
+-----+
| information_schema |
| mysql      |
| performance_schema |
| sakila     |
| sys       |
| test      |
| world     |
+-----+
7 rows in set (0.00 sec)

mysql>
```

Рисунок 9 – Создание собственной базы данных

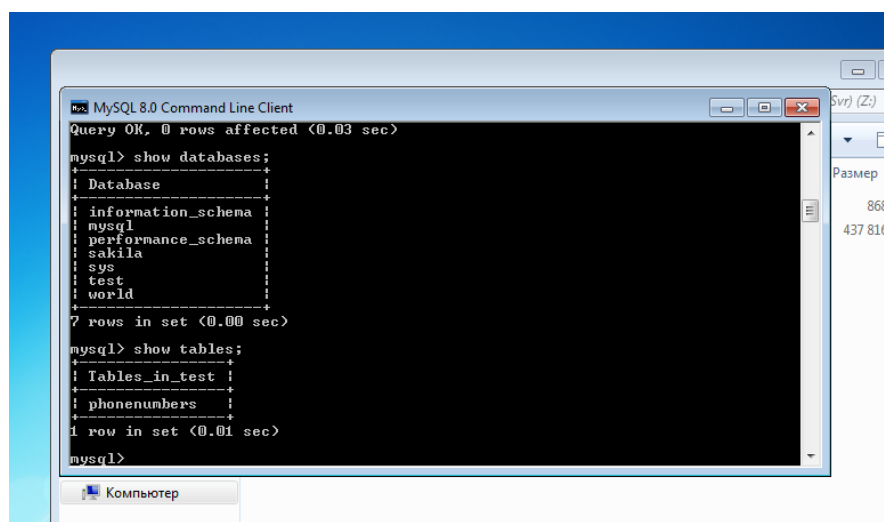
Необходимо указать серверу, что далее работа будет происходить именно с ней, используя следующую команду use test;



```
mysql> use test;
Database changed

mysql>
```

Рисунок 10 – Создание используемой БД



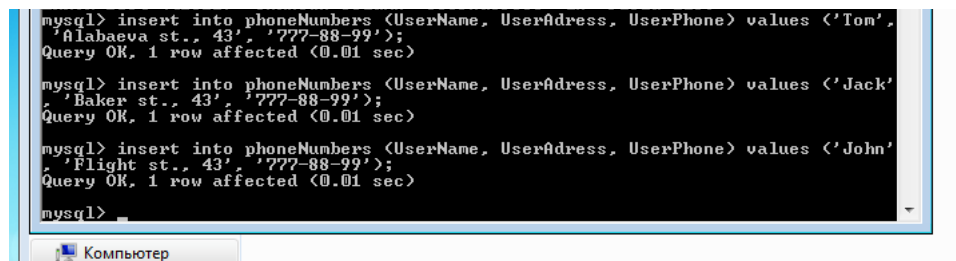
```
mysql> show databases;
+-----+
| Database |
+-----+
| information_schema |
| mysql      |
| performance_schema |
| sakila     |
| sys       |
| test      |
| world     |
+-----+
7 rows in set (0.00 sec)

mysql> show tables;
+-----+
| Tables_in_test |
+-----+
| phonenumbers   |
+-----+
1 row in set (0.01 sec)

mysql>
```

Рисунок 11 - Проверка создания

Поскольку сейчас таблица пустая, заполним ее произвольными данными – в нашем примере это телефонная книга, поэтому добавим в нее записи, содержащие имя, адрес и телефонный номер абонентов с помощью команды `insert into phoneNumbers (UserName, UserAddress, UserPhone) values ('Tom', 'Alabaeva st., 43', '777-88-99');`



```
mysql> insert into phoneNumbers (UserName, UserAddress, UserPhone) values ('Tom',
'Alabaeva st., 43', '777-88-99');
Query OK, 1 row affected (0.01 sec)

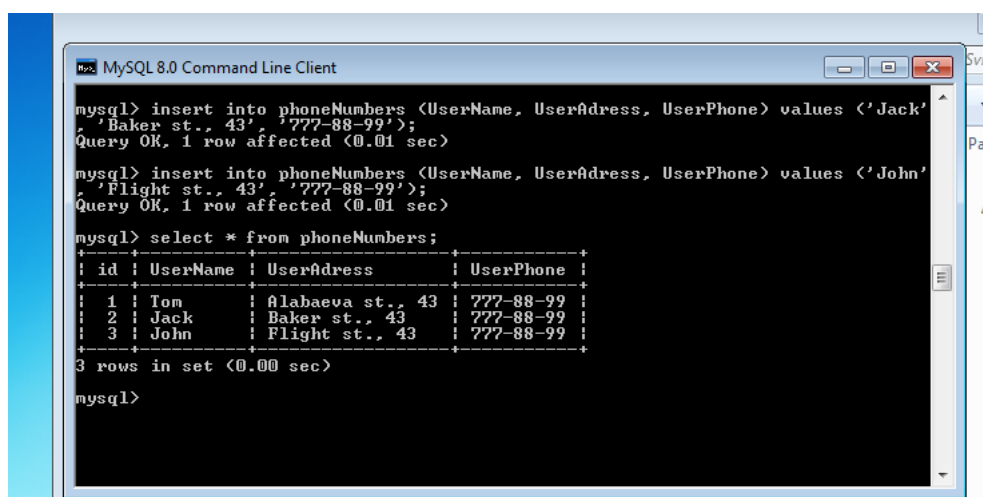
mysql> insert into phoneNumbers (UserName, UserAddress, UserPhone) values ('Jack'
'Baker st., 43', '777-88-99');
Query OK, 1 row affected (0.01 sec)

mysql> insert into phoneNumbers (UserName, UserAddress, UserPhone) values ('John'
'Flight st., 43', '777-88-99');
Query OK, 1 row affected (0.01 sec)

mysql>
```

Рисунок 12 - Добавление записей

Посмотрим содержимое таблицы. Поскольку команды с `show` обращаются к файловой системе, необходимо извлечь все столбцы и строки созданной нами таблицы `phoneNumbers`, что можно сделать следующей командой: `select * from phoneNumbers;`



```
mysql> insert into phoneNumbers (UserName, UserAddress, UserPhone) values ('Jack'
'Baker st., 43', '777-88-99');
Query OK, 1 row affected (0.01 sec)

mysql> insert into phoneNumbers (UserName, UserAddress, UserPhone) values ('John'
'Flight st., 43', '777-88-99');
Query OK, 1 row affected (0.01 sec)

mysql> select * from phoneNumbers;
+----+-----+-----+-----+
| id | UserName | UserAddress | UserPhone |
+----+-----+-----+-----+
| 1  | Tom      | Alabaeva st., 43 | 777-88-99 |
| 2  | Jack     | Baker st., 43   | 777-88-99 |
| 3  | John     | Flight st., 43  | 777-88-99 |
+----+-----+-----+-----+
3 rows in set (0.00 sec)

mysql>
```

Рисунок 13 – Просмотр содержимого таблицы `phoneNumbers`

#### 4) Создание нового пользователя и настройка его прав доступа

После первоначальной настройки сервера MySQL, будет предоставлено имя пользователя и пароль, причем эти начальные учётные данные дают привилегии «root-доступа», то есть привилегированного пользователя. Пользователь с правами доступа `root` имеют полный доступ ко всем базам данных и таблицам внутри этих баз – но на предприятиях обычно требуется предоставить доступ к базе данных для работников, которым не требуется и не рекомендовано полное управление.

Для создания нового пользователя следует выполнить следующие шаги:

- 1) Создать пользователя MySQL и предоставить неограниченные права доступа

## 2) Назначить специальные права доступа для пользователя MySQL

Создание пользователя MySQL и предоставление ему неограниченных прав доступа

Создадим пользователя с именем LocalUser и установим ему пароль password. По заданию требуется создать локального пользователя – пользователя, который будет существовать на текущей АРМ. Это логично с точки зрения безопасности: нет необходимости создавать глобального пользователя, который будет иметь возможность войти в свою учетную запись с любого АРМ – в этом случае требовалось бы устанавливать какие-то дополнительные политики безопасности на каждом АРМ, с которого можно осуществить работу. В нашем случае достаточно будет настроить, скажем, локальные политики безопасности (см. лаб. раб. 1) и ограничить пользователю доступ в интернет только одним IP адресом – тем, на котором работает сервер MySQL.

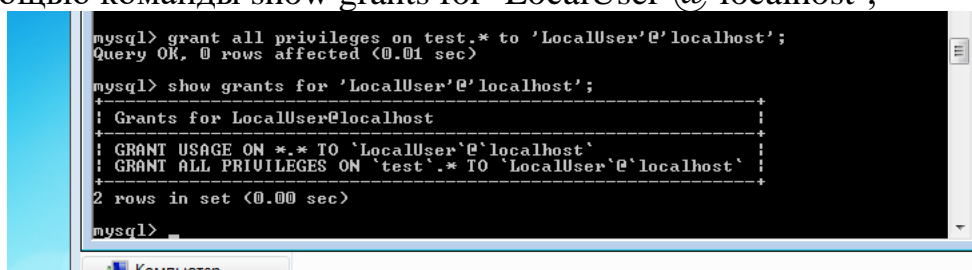
Для этого выполним следующую команду create user 'LocalUser'@'localhost' identified by 'password';

```
mysql> create user 'LocalUser'@'localhost' identified by 'password';  
Query OK, 0 rows affected (0.09 sec)
```

Рисунок 14 – Создание нового пользователя

Теперь необходимо назначить для созданного пользователя права доступа, сначала следует предоставить ему все права, а потом настроить в соответствии с политикой безопасности предприятия и регламентом работы с базами данных. Чтобы назначить вновь созданному пользователю неограниченные права доступа к нашей пользовательской базе данных test, выполним следующую команду grant all privileges on test.\* to 'LocalUser'@'localhost';

Теперь проверим, применились ли права, просмотрев их для пользователя с помощью команды show grants for 'LocalUser'@'localhost';



```
mysql> grant all privileges on test.* to 'LocalUser'@'localhost';  
Query OK, 0 rows affected (0.01 sec)  
  
mysql> show grants for 'LocalUser'@'localhost';  
+-----+  
| Grants for LocalUser@localhost |  
+-----+  
| GRANT USAGE ON *.* TO 'LocalUser'@'localhost' |  
| GRANT ALL PRIVILEGES ON `test`.* TO 'LocalUser'@'localhost' |  
+-----+  
2 rows in set (0.00 sec)  
  
mysql>
```

Рисунок 15 – Добавление пользователя

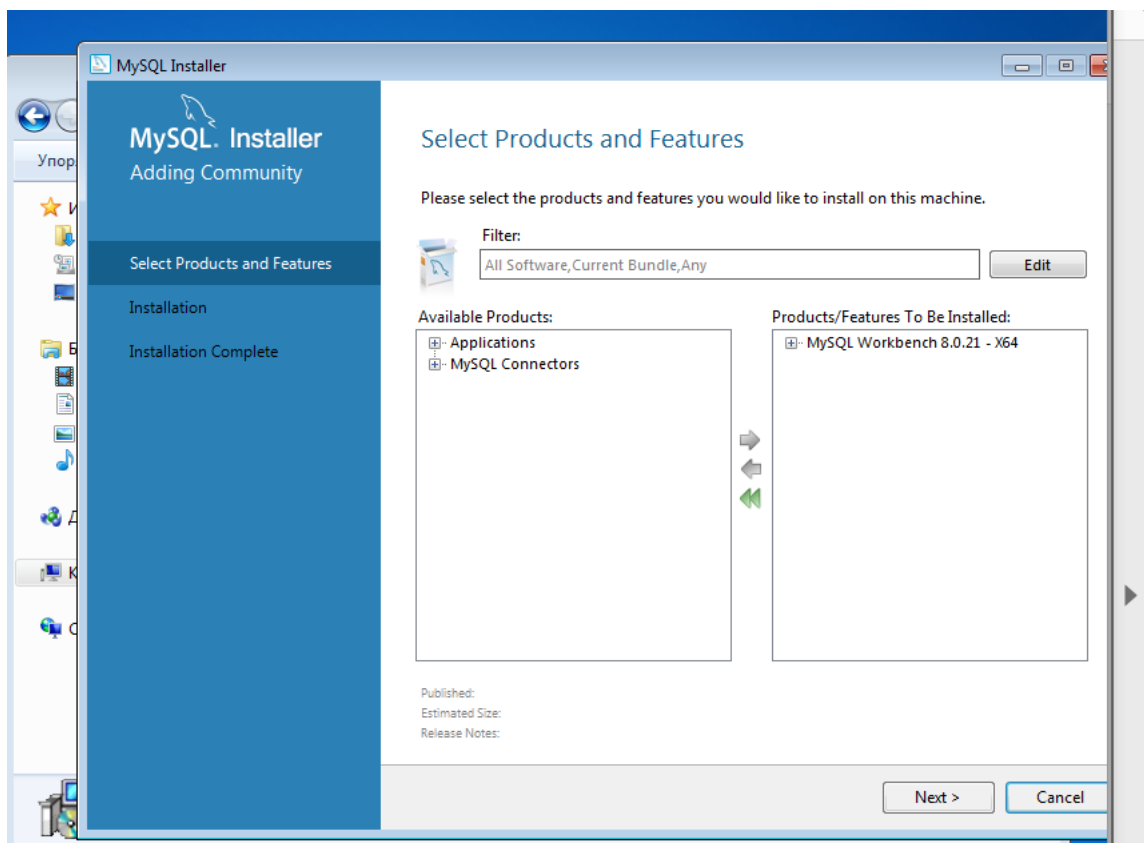


Рисунок 16 – Установка «MySQL Workbench»

Чтобы запустить командную строку под новым пользователем необходимо зайти в интерфейс программы «MySQL Workbench» и создать новое подключение, вызвав *Manage Server Connections*, для которого настроить следующие параметры, а остальные оставить по умолчанию:

- Connection Name: LocalUser
- Password: password

После настройки *Manage Server Connections* должен выглядеть следующим образом (рис.5.14):

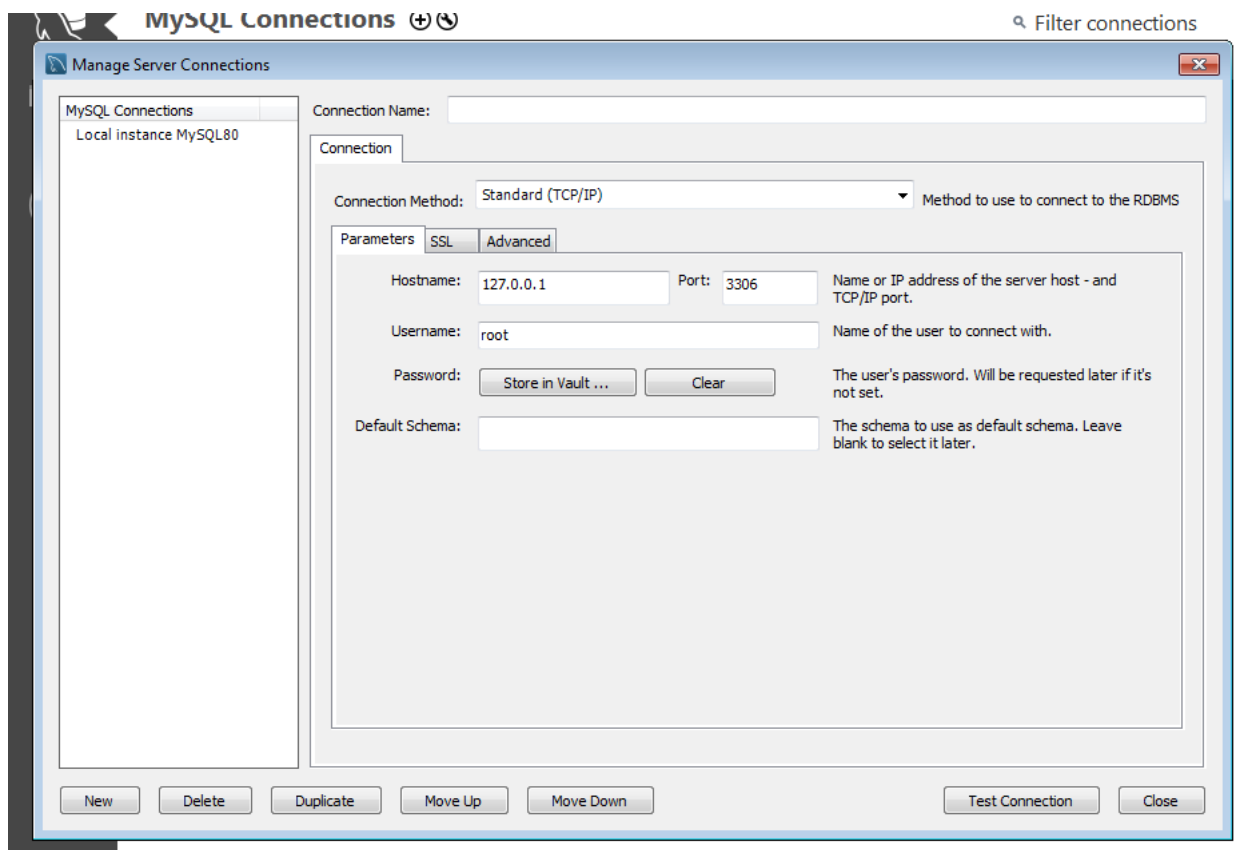


Рисунок 17 - Добавление нового соединения

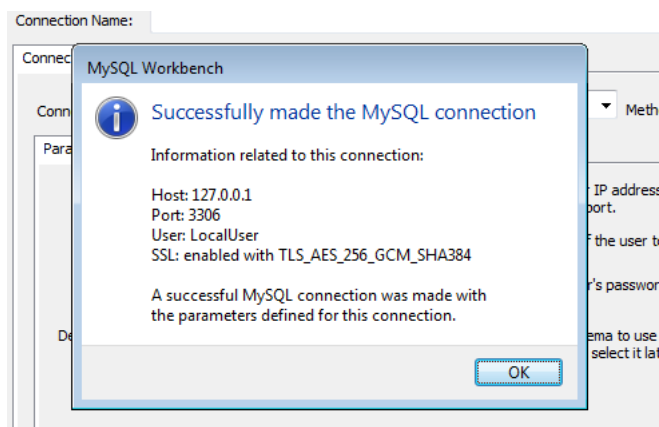


Рисунок 18 – Удачный результат подключения

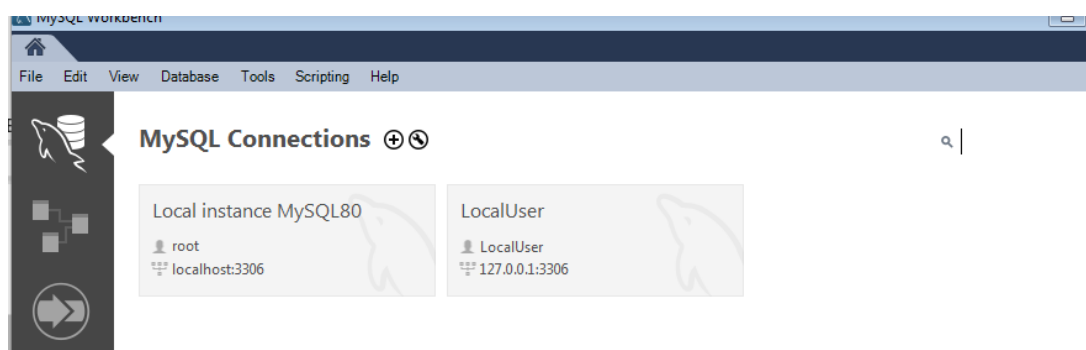


Рисунок 19 – Отображение нового соединения на главной странице

Попробуем подключиться к серверу с использованием этого нового соединения, выполнив следующие действия: *Кликнуть на подключение LocalUser правой кнопкой мыши → Start Command Line Client*  
Как можно заметить, подключение прошло успешно

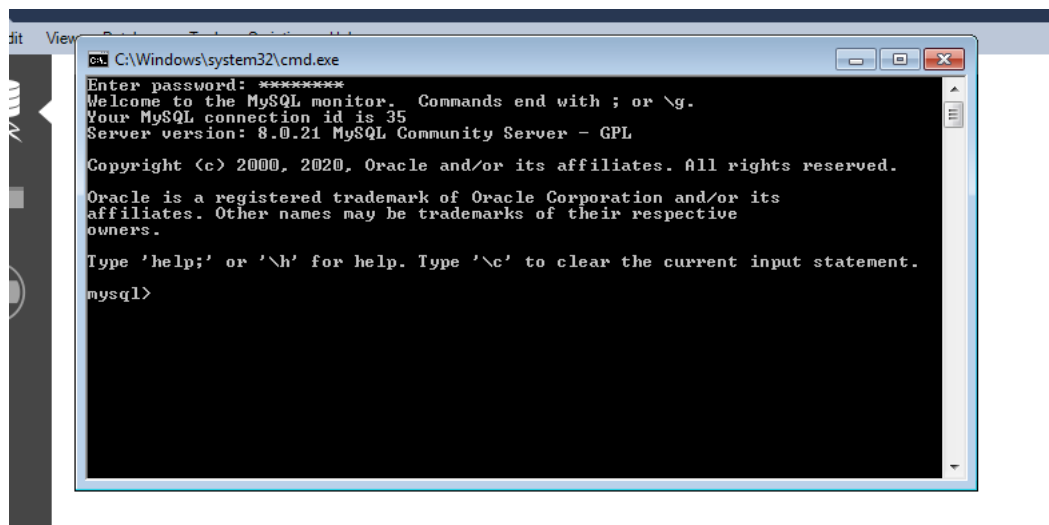


Рисунок 20 – Успешное подключение пользователя LocalUser к серверу

Очевидно, что не привилегированный пользователь успешно получил доступ к серверу с помощью утилиты «MySQL Workbench».

#### Назначение специальных прав доступа для пользователя MySQL

В процессе настроек сервера «MySQL Server» была пропущена возможность создания нового пользователя с заданными правами. Используем ее при выполнении данного этапа задания. MySQL позволяет назначать права доступа с помощью следующей команды:

**GRANT [тип прав] ON [имя базы данных].[имя таблицы] TO 'имя пользователя'@'тип доступа на сервер';**

Нужно заменить значение «тип прав» на тот вид прав доступа, который вы хотите предоставить новому пользователю. Также вам нужно указать базу данных и имена таблиц, доступ к которым предоставляется. В MySQL есть несколько типов прав доступа, некоторые из них описаны ниже:

- **CREATE** – Позволяет пользователям создавать базы данных/таблицы
- **SELECT** – Позволяет пользователям делать выборку данных
- **INSERT** – Позволяет пользователям добавлять новые записи в таблицы
- **UPDATE** – Позволяет пользователям изменять существующие записи в таблицах
- **DELETE** – Позволяет пользователям удалять записи из таблиц
- **DROP** – Позволяет пользователям удалять записи в базе данных/таблицах

Разрешим пользователю только просмотр записей остальных баз данных, имеющихся в файловой системе SQL: их изменение недопустимо для



обычного непривилегированного пользователя, потому что может привести к искажению и даже потере важных данных, как уже упоминалось выше. Сделаем это следующей командой: `grant SELECT on *.* to 'LocalUser'@'localhost';`

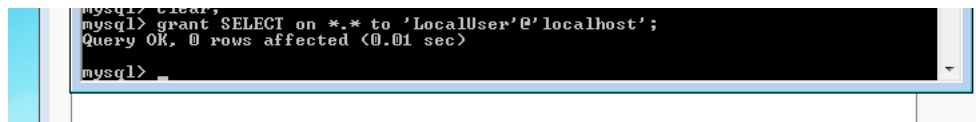


Рисунок 21 -Назначение прав пользователю LocalUser

Предположим, что АРМ, за которым работает пользователь, расположено в «закрытом» контуре и на нем обрабатываются важные данные. Поскольку в предыдущем задании была создана таблица-телефонный справочник, логично предположить, что пользователь АРМ оперирует данными, предположим, клиентов предприятия. Следовательно, ему можно разрешить права **SELECT**, делать выборку из всех записей таблицы (с целью поиска определенного человека, например, и уточнения его личных данных), и **INSERT**, право на добавление новых записей (если потребуется клиентскую базу). Однако, пользователю нельзя разрешать создавать новые таблицы, модифицировать уже имеющиеся записи и тем более удалять записи и таблицы из базы данных – это может привести к потере важных данных и, как следствие, к потере ресурсов. В случае, если пользователем будет совершена ошибка, ему следует обратиться к администратору (root) базы данных предприятия.

Установим права **SELECT** и **INSERT** на базу данных test и непосредственно для таблицы phoneNumbers для пользователя LocalUser следующими командами:

```
grant SELECT, INSERT on test.* to 'LocalUser'@'localhost';  
grant SELECT, INSERT on laboratoryWork3.phoneNumbers to  
'LocalUser'@'localhost';
```

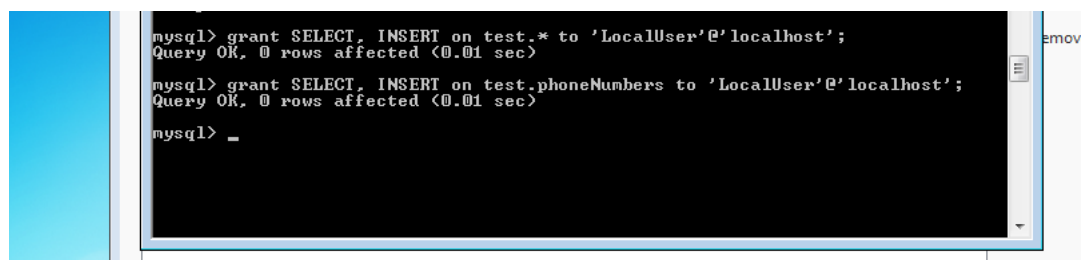
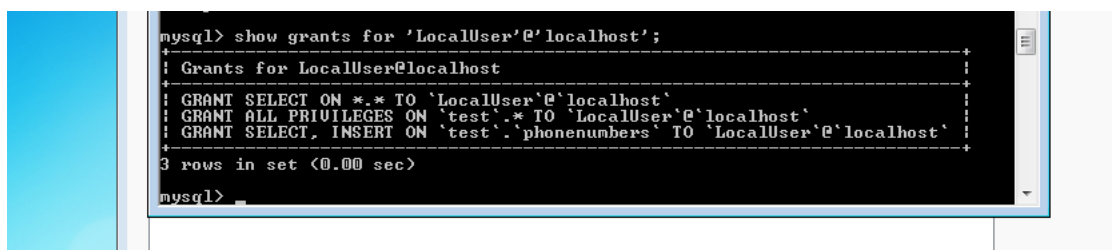


Рисунок 22 – Назначение прав пользователю LocalUser

Рис. 5.18. Назначение прав пользователю LocalUser

В таблице phoneNumbers пользователь LocalUser имеет право просматривать любые поля и записи, поскольку в соответствии с его ролью на предприятии он оперирует этими данными, поэтому ограничения на доступ к отдельным полям таблицы устанавливать не нужно.

Теперь проверим, применились ли права, просмотрев их для пользователя с помощью команды: `show grants for 'LocalUser'@'localhost';`



```
mysql> show grants for 'LocalUser'@'localhost';
+-----+
| Grants for LocalUser@localhost |
+-----+
| GRANT SELECT ON *.* TO 'LocalUser'@'localhost' |
| GRANT ALL PRIVILEGES ON 'test'.* TO 'LocalUser'@'localhost' |
| GRANT SELECT, INSERT ON 'test'.phoneNumbers TO 'LocalUser'@'localhost' |
+-----+
3 rows in set (0.00 sec)

mysql>
```

Рисунок 23 – Просмотр прав пользователя LocalUser

Очевидно, что изменение прав доступа к базе данных и содержащимся в них таблицам для пользователя LocalUser успешно осуществлено в соответствии с ролью пользователя на предприятии.

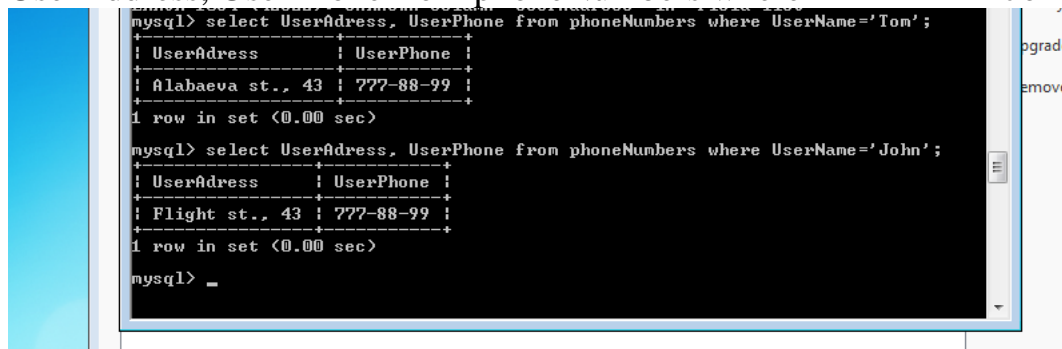
#### Запрос выборок из таблицы

Проверим корректность настроенных прав для пользователя LocalUser, а также ознакомимся с механизмом запроса записей из таблицы в SQL.

Сделаем несколько выборок из таблицы phoneNumbers:

Выборка из таблицы phoneNumbers значений адреса и телефона для пользователей 'Tom' и 'Edvard' с помощью следующих команд:

`select UserAddress, UserPhone from phoneNumbers where UserName='Tom';`  
`select UserAddress, UserPhone from phoneNumbers where UserName='John';`



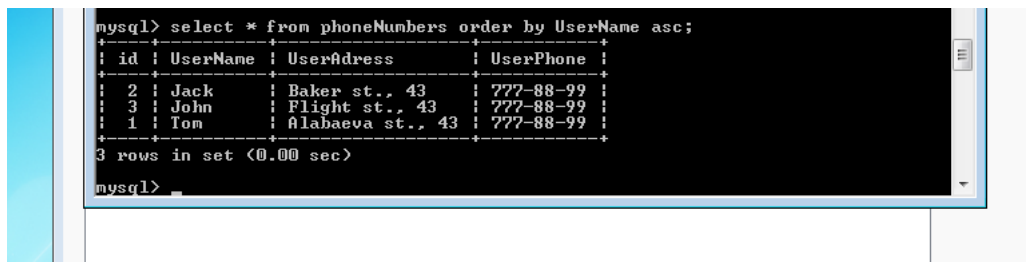
```
mysql> select UserAddress, UserPhone from phoneNumbers where UserName='Tom';
+-----+-----+
| UserAddress | UserPhone |
+-----+-----+
| Alabaeva st., 43 | 777-88-99 |
+-----+-----+
1 row in set (0.00 sec)

mysql> select UserAddress, UserPhone from phoneNumbers where UserName='John';
+-----+-----+
| UserAddress | UserPhone |
+-----+-----+
| Flight st., 43 | 777-88-99 |
+-----+-----+
1 row in set (0.00 sec)

mysql>
```

Рисунок 24 – Результат выборки по значениям адреса и телефона для двух произвольных пользователей

Выборка всех записей из таблицы phoneNumbers с сортировкой по полю UserName в алфавитном порядке с использованием следующей команды:  
`select * from phoneNumbers order by UserName asc;`

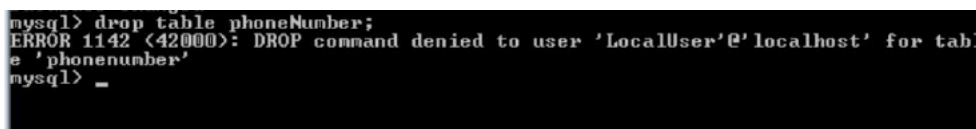


```
mysql> select * from phoneNumbers order by UserName asc;
+----+-----+-----+-----+
| id | UserName | UserAddress | UserPhone |
+----+-----+-----+-----+
| 2  | Jack    | Baker st., 43 | 777-88-99 |
| 3  | John    | Flight st., 43 | 777-88-99 |
| 1  | Tom     | Alabaeva st., 43 | 777-88-99 |
+----+-----+-----+-----+
3 rows in set (0.00 sec)

mysql> _
```

Рисунок 25 – Результат выборки по всем записям в алфавитном порядке

Проверим, корректно ли работает механизм задания прав пользователя, попробовав удалить таблицу phoneNumbers с использованием следующей команды drop table phoneNumbers;



```
mysql> drop table phoneNumber;
ERROR 1142 (42000): DROP command denied to user 'LocalUser'@'localhost' for table 'phonenumber'
mysql> _
```

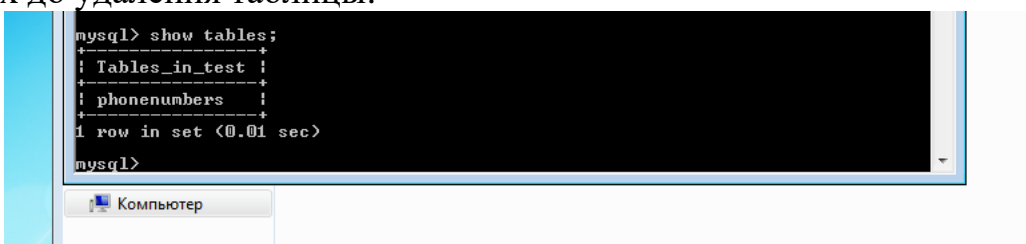
Рисунок 26 – Попытка удаления таблицы пользователем LocalUser, у которого нет на это прав

Очевидно, что пользователю LocalUser действительно не хватает прав на удаление таблицы phoneNumbers – как и настраивалось ранее.

### Удаление таблиц, баз данных и пользователей

На последнем этапе рассмотрим механизм удаления таблиц, баз данных и пользователей из файловой системы SQL. Это можно сделать только привилегированным-root пользователем, поэтому дальнейшая работа будет осуществляться из его терминала.

1) Удаление таблицы phoneNumbers из базы данных test. Содержимое базы данных до удаления таблицы:



```
mysql> show tables;
+-----+
| Tables_in_test |
+-----+
| phonenumber     |
+-----+
1 row in set (0.01 sec)

mysql>
```

Рисунок 27 – Содержимое БД до удаления таблицы phoneNumbers

Удалим таблицу phoneNumbers с помощью команды: drop table phoneNumbers ;



```
mysql> drop table phoneNumbers;
Query OK, 0 rows affected (0.03 sec)
```

Рисунок 28 -Удаление таблицы



Рисунок 29 – Содержимое БД после удаление таблицы phoneNumbers

- 2) Удаление базы данных test при помощи команды drop database test;.  
Список баз данных до удаления и после удаления.

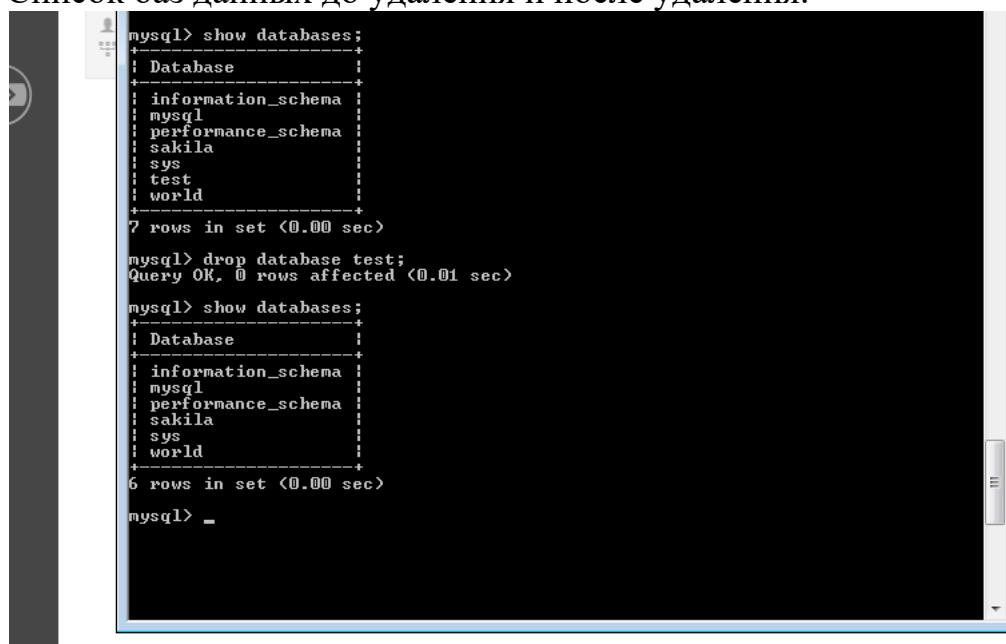


Рисунок 30 – Удаление БД

- 3) Удаление всех прав доступа пользователя LocalUser с помощью следующей команды: revoke all privileges on \*.\* from 'LocalUser'@'localhost';

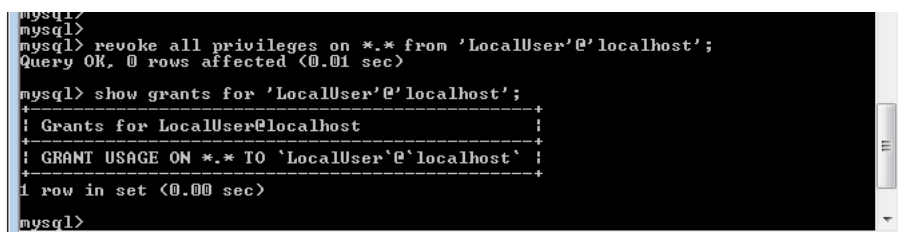
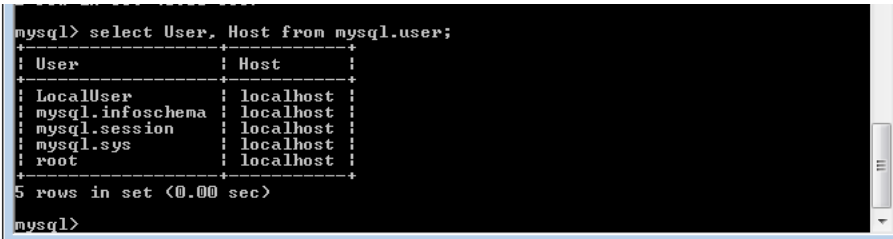


Рисунок 31 – Удаление прав пользователя LocalUser и проверка корректности выполнения команды

- Удалим пользователя LocalUser. Посмотрим список всех пользователей до удаления LocalUser, обратившись к таблице mysql.user и запросив из нее всех

пользователей и их способ подключения к серверу с помощью команды: select User, Host from mysql.user;

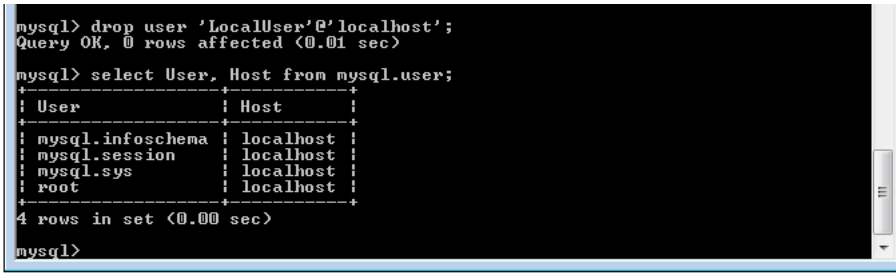


```
mysql> select User, Host from mysql.user;
+-----+-----+
| User           | Host       |
+-----+-----+
| LocalUser      | localhost  |
| mysql.infoschema | localhost  |
| mysql.session  | localhost  |
| mysql.sys      | localhost  |
| root           | localhost  |
+-----+-----+
5 rows in set (0.00 sec)

mysql>
```

Рисунок 32 – Список имеющихся пользователей до удаления LocalUser

Удалим пользователя LocalUser с помощью команды drop user 'LocalUser'@'localhost';



```
mysql> drop user 'LocalUser'@'localhost';
Query OK, 0 rows affected (0.01 sec)

mysql> select User, Host from mysql.user;
+-----+-----+
| User           | Host       |
+-----+-----+
| mysql.infoschema | localhost  |
| mysql.session    | localhost  |
| mysql.sys        | localhost  |
| root            | localhost  |
+-----+-----+
4 rows in set (0.00 sec)

mysql>
```

Рисунок 33 – Список имеющихся пользователей после удаления LocalUser

Очевидно, что удаление баз данных, пользователей и таблиц успешно осуществлено.

#### 4. Выводы.

В работе проведена установка MySQL, создана база данных. Назначены и проверены права пользователя по доступу к ресурсам сервера базы данных. Настройка параметров безопасности с учетом принятой политики информационной безопасности в организации - прав доступа к ресурсам сервера базы данных, а именно, SELECT, права на выборку из всех записей таблицы (с целью поиска определенного человека, например, и уточнения его личных данных), и INSERT, право на добавление новых записей (если потребуется клиентскую базу) позволяет повысить уровень защиты базы данных от НСД.