Семинар 5

Основные характеристики MySQL

Язык SQL

(структурированный язык запросов)

Язык SQL предназначен для манипуляции данными, хранящихся в РБД. SQL имеет команды, с помощью которых можно создавать объекты БД, манипулировать ими, осуществлять начальную загрузку данных, извлекать, сортировать, обновлять, удалять и добавлять данные .

Команды SQL можно разделить на категории:

1. DDL, язык определения данных (*CREATE (ALTER, DROP) TABLE, INDEX*) ;
2. DML, язык манипулирования данными (*INSERT,UPDATE,DELETE*);
3. DQL, язык запросов *(SELECT)*;
4. DCL, язык управления данными (*доступ к информации в БД*);
5. команды администрирования данных;
6. Команды управления транзакциями;

Стандарты языка SQL определяет ANSI . В настоящее время действует стандарт, принятый в 2003 году (SQL-3).

SQL можно использовать с такими РСУБД как MySQL, mSQL, Oracle, Линтор, Microsoft SQL Server, Access, Sybase и др.
Эти системы поддерживают все общепринятые операторы SQL

Реляционная СУБД MySQL

MySQL —реляционная *СУБД,* открыто распространяемая. ПО MySQL является системой клиент-сервер, содержит многопоточный SQL-сервер, для обеспечения поддержки различных вычислительных машин БД, несколько различных клиентских программ и библиотек, средства администрирования и программные интерфейсы (API).

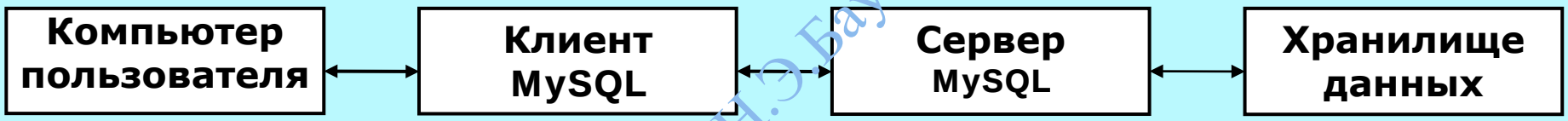


Рис. 1.3 Схема передачи данных в архитектуре "клиент/сервер"

Сервер *ресурса в компьютерной сети – компьютер (программа), управляющий ресурсом;*

клиент *– компьютер (программа), использующий этот ресурс.*

Достоинство организации ИС по архитектуре клиент-сервер – сочетание централизованного хранения, обслуживания и коллективного доступа к общей корпоративной информации с индивидуальной работой пользователей над персональной информацией.

Реляционная СУБД MySQL

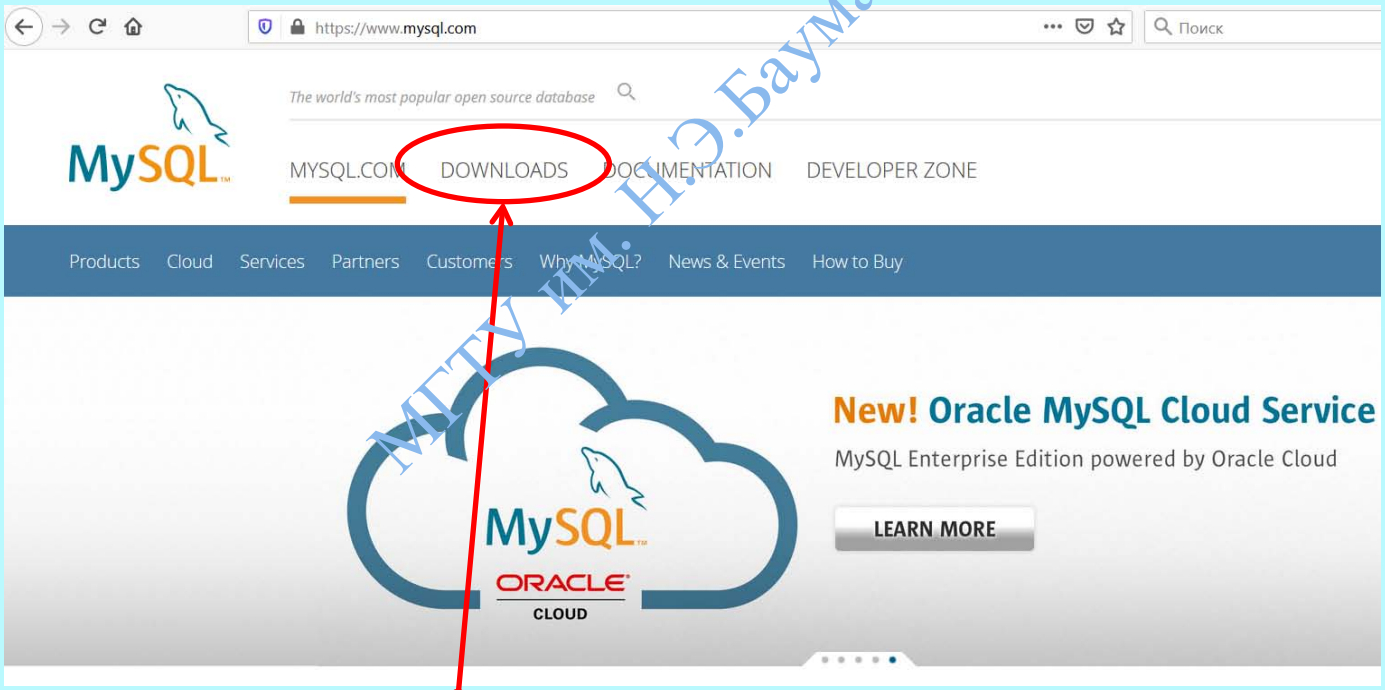
Сайт разработчика

https://www.mysql.com

Установка MySQL

Есть несколько вариантов установки MySQL.

1. Установка MySQL Server

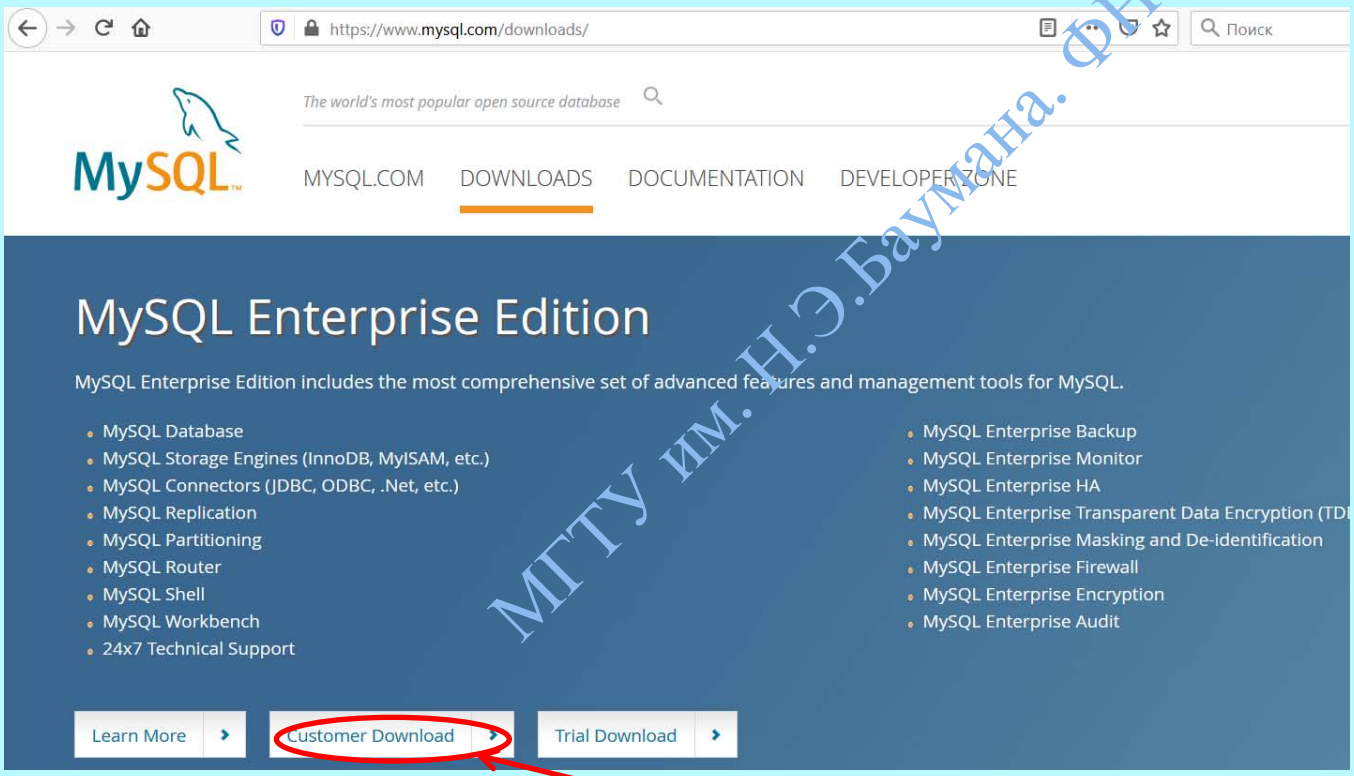


1. Открыть вкладку **DOWNLOADS**

Реляционная СУБД MySQL

Сайт разработчика

https://www.mysql.com /downloads/

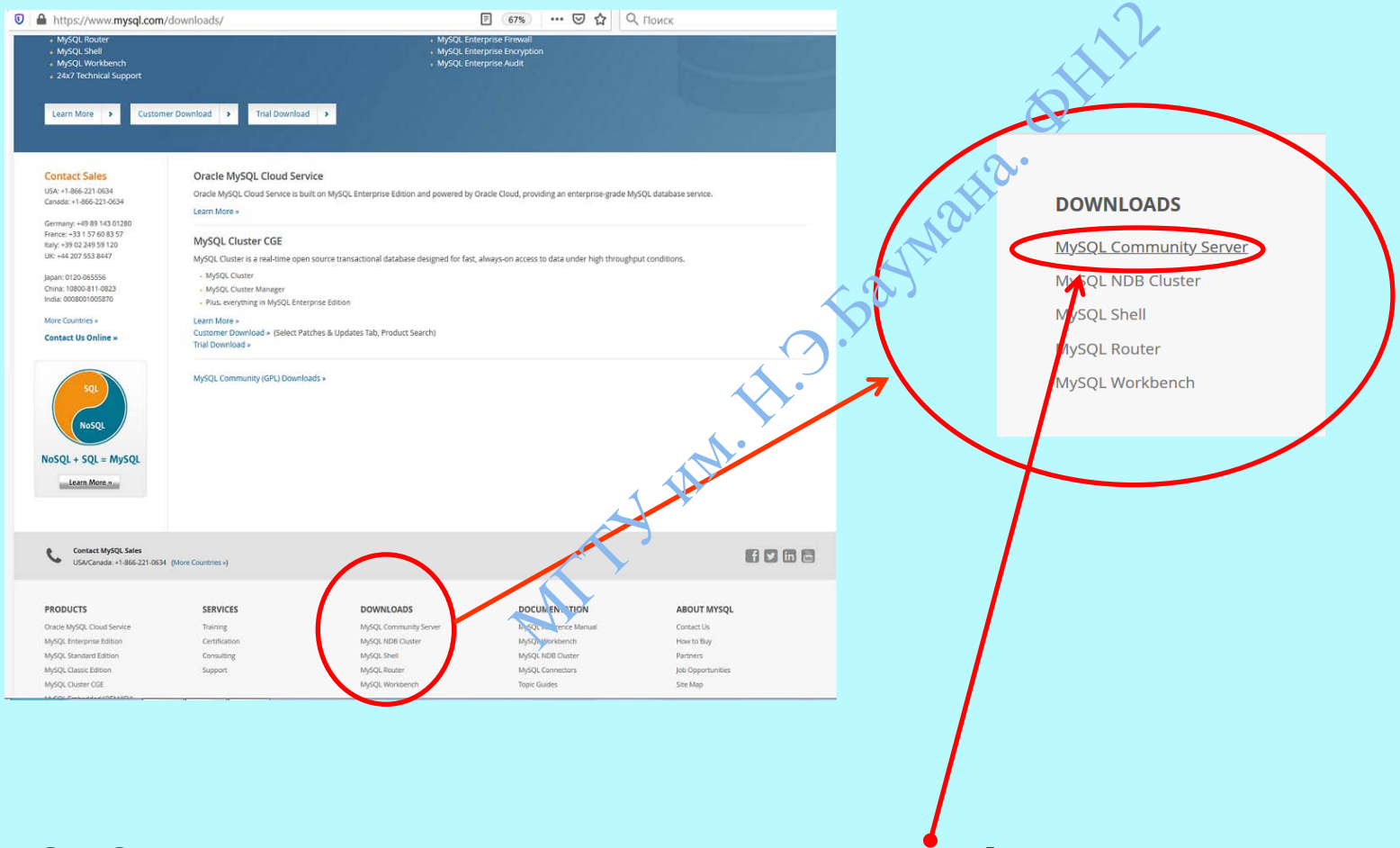


2. Открыть вкладку **CustomerDownloads.**

Реляционная СУБД MySQL

Сайт разработчика

https://www.mysql.com /downloads/

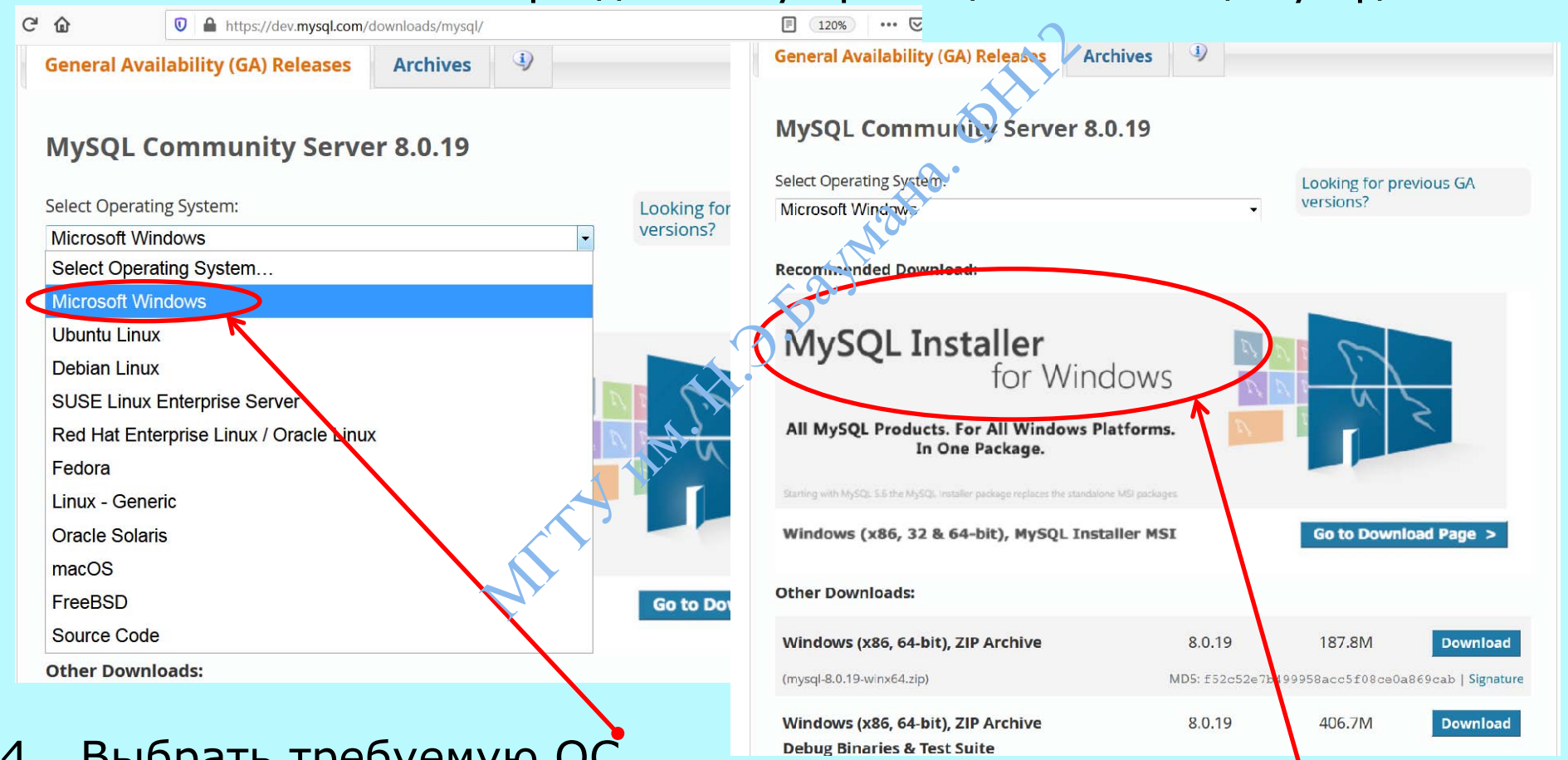


3. Открыть вкладку **MySQL Community Server**

Реляционная СУБД MySQL

Сайт

https://dev.mysql.com/downloads/mysql/

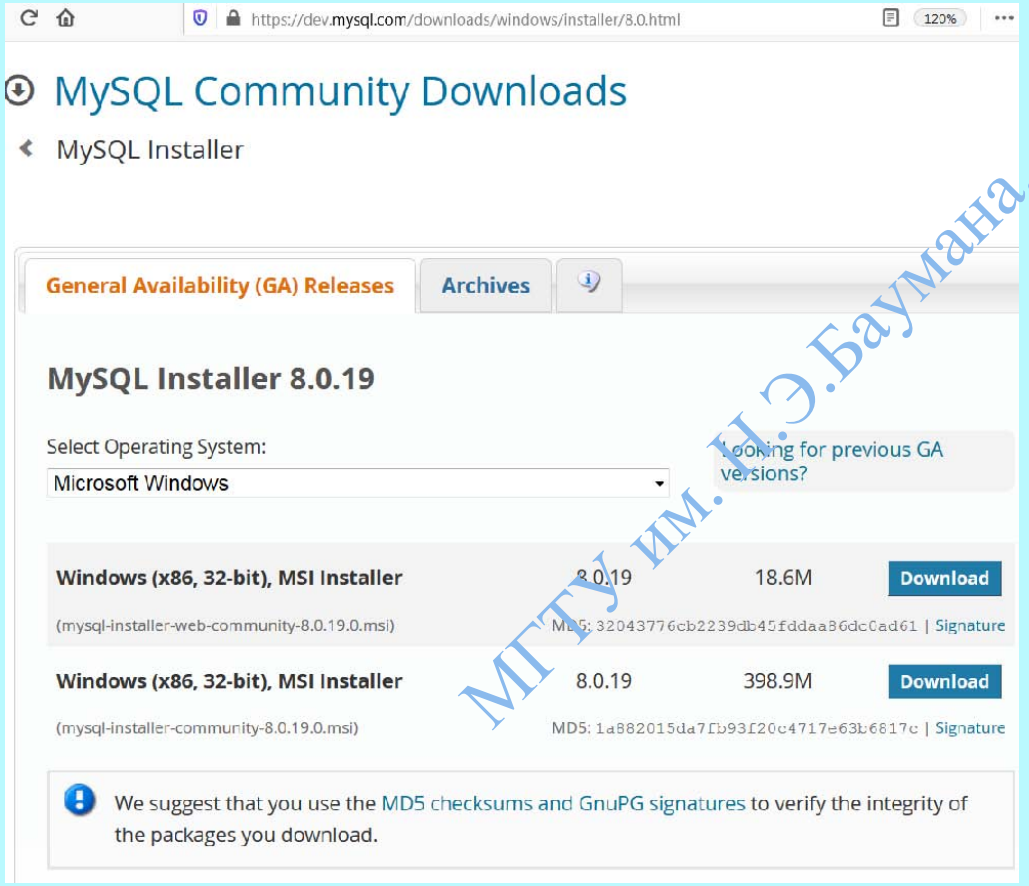


4. Выбрать требуемую ОС.
5. Скачать автоматический установщик MySQL Insteller

Реляционная СУБД MySQL

Сайт

https://dev.mysql.com/downloads/mysql/



6. Выбрать нужный **MSI Installer**

Реляционная СУБД MySQL

Сайт

https://dev.mysql.com/downloads/mysql/

6. Для скачивания Установщика можно зарегистрироваться для получения бесплатной учетной или отказаться и скачать без регистрации.

7. Скачать файл *mysql-installer-web-community-8.0.19.0*

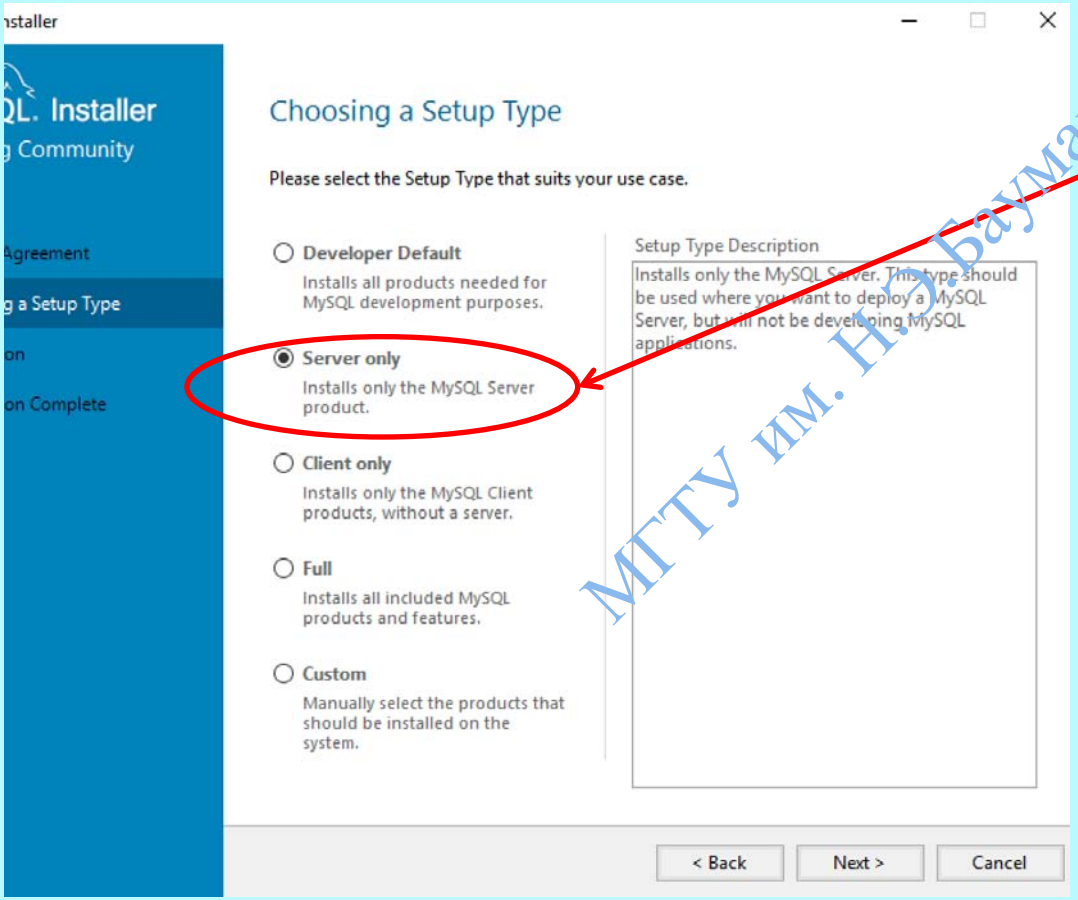
В файле *mysql-installer* есть все необходимое для установки.

Реляционная СУБД MySQL

Установка СУБД MySQL .

Запустить файл *mysql-installer-web-community-8.0.19.0* и

1. Принять условия лицензионного соглашения, нажать **Next**.



2. Выбирать тип установки
3. Выбрать установку только сервера или Developer (вариант по умолчанию).
4. Нажать Next.

*При выборе варианта **Developer** отдельно устанавливать Workbench не требуется .*

Реляционная СУБД MySQL

Установка СУБД MySQL .

Варианты установки.

Developer –установка всех инструментов для работы с MySQL требуемых разработчику (MySQL Server, MySQL Workbench и др.) (по умолчанию) ;

Server Only –установка только сервера MySQL;

Client Only –установка только клиентской части для работы с MySQL Server, без серверной части;

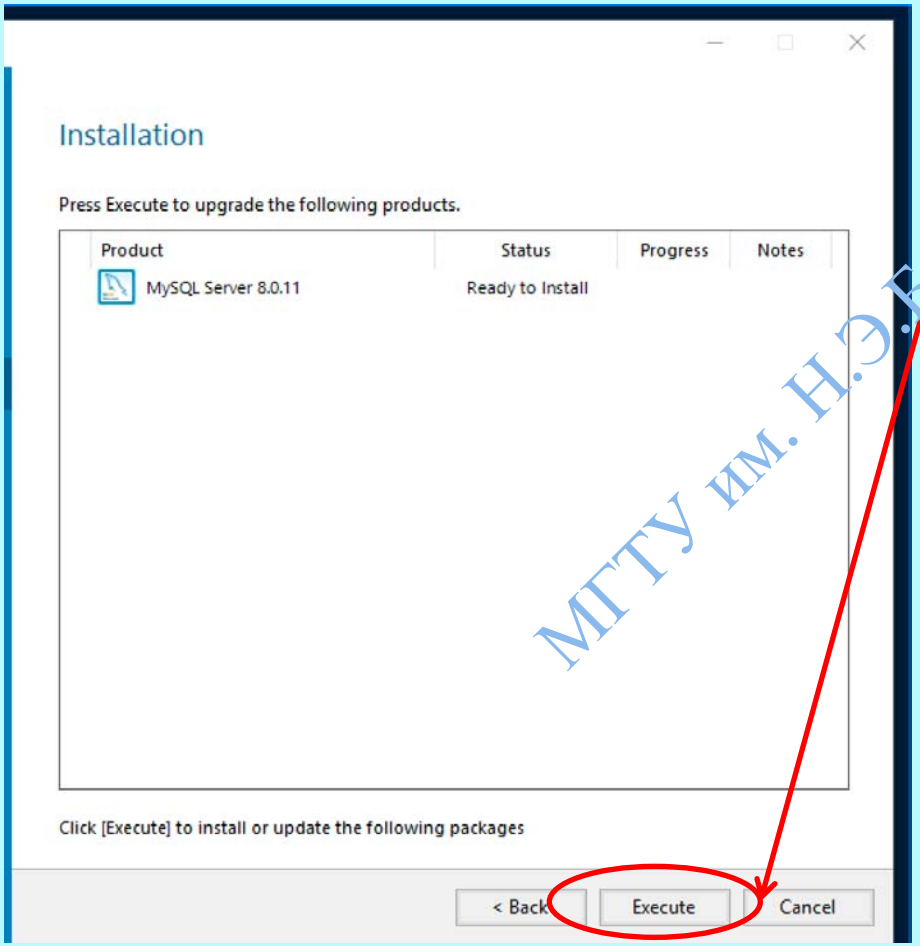
Full – установка всех включённых в дистрибутив компонентов;

Custom – выборочная установка, можно установить только требуемые инструменты.

Реляционная СУБД MySQL

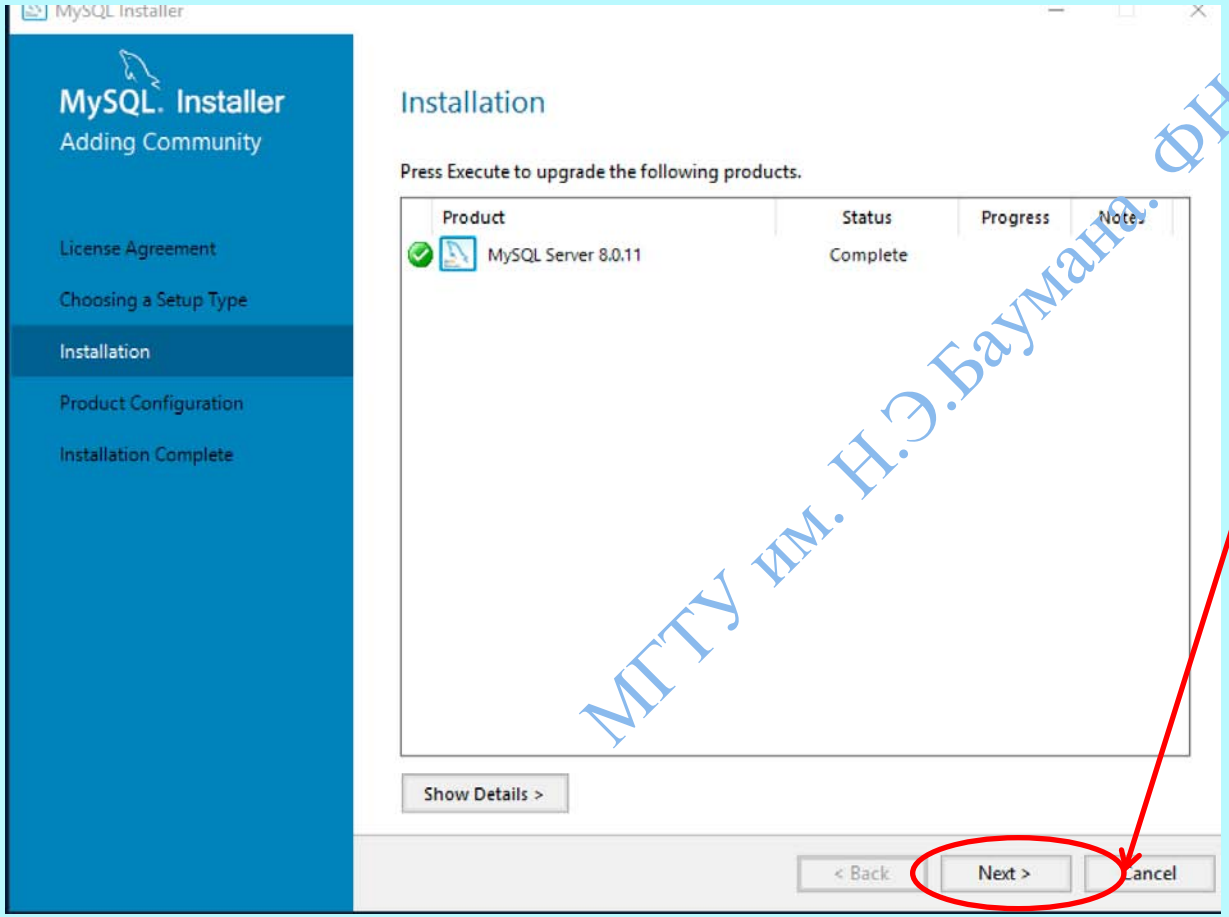
Сайт

https://dev.mysql.com/downloads/mysql/



5. Нажать **Execute** и ждать завершения установки.

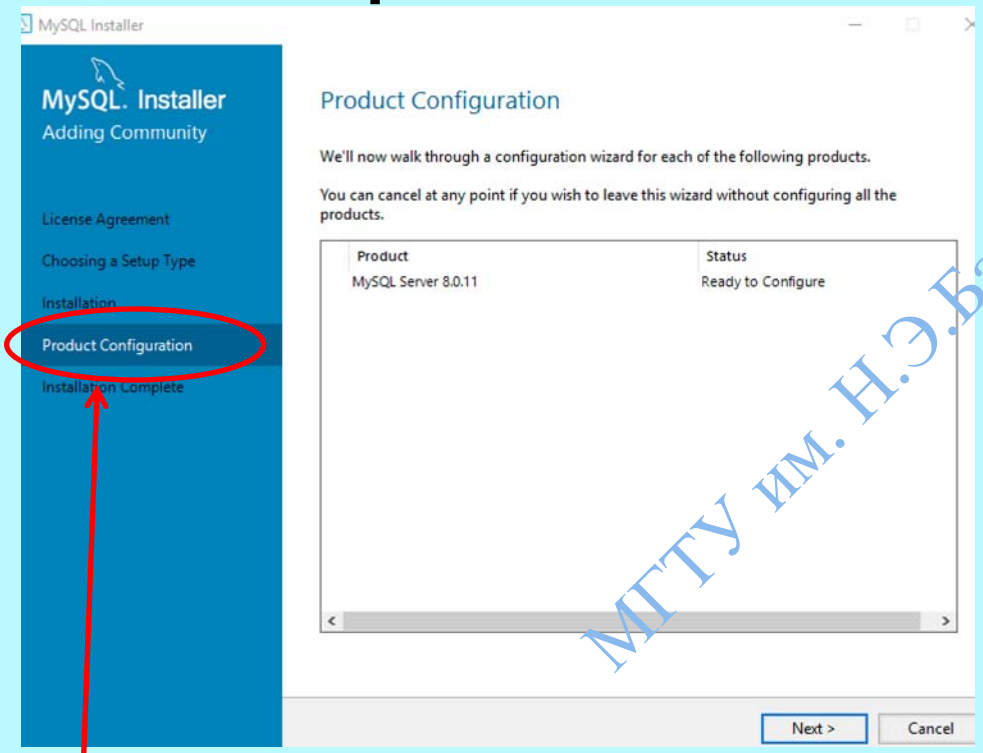
Реляционная СУБД MySQL

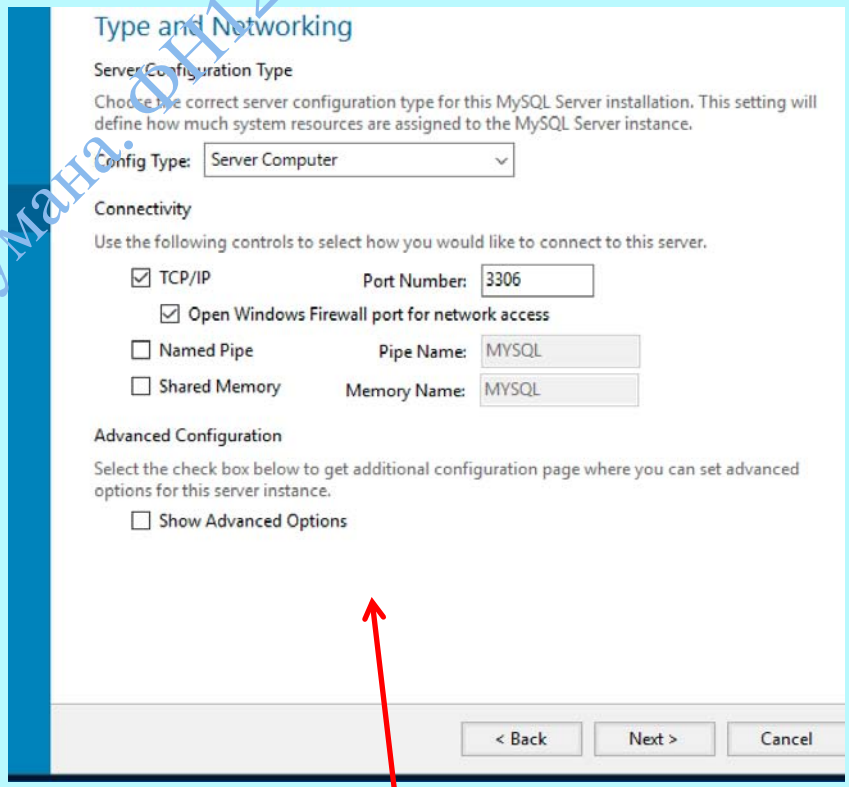


6. Нажать **Next.**

Реляционная СУБД MySQL

Этап настройки

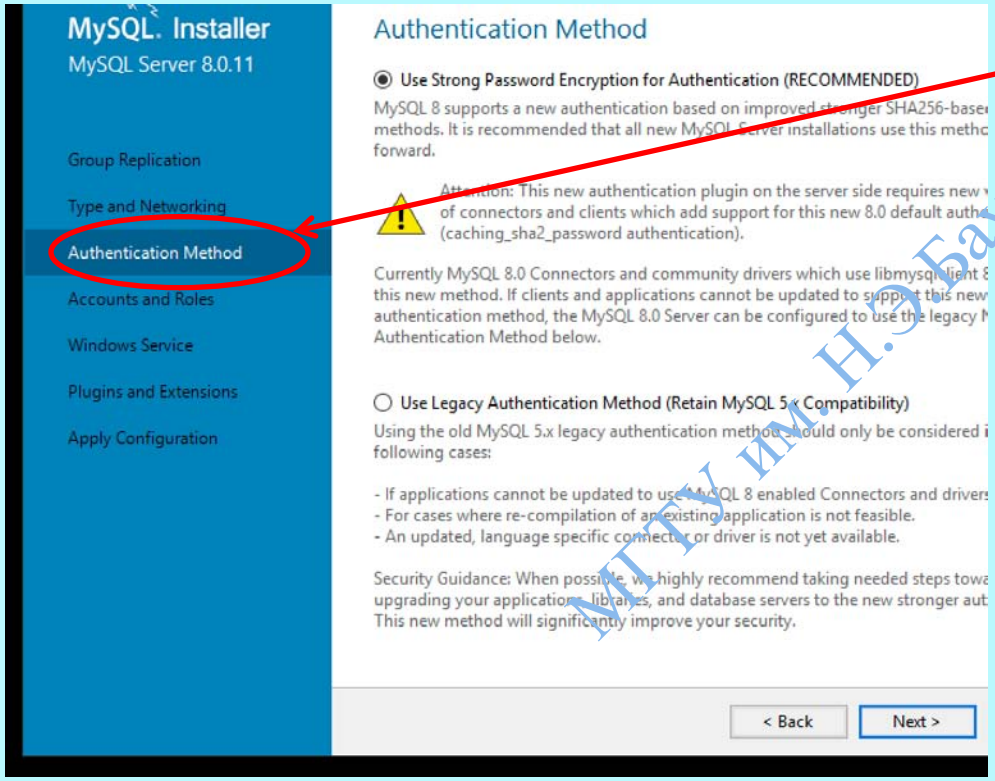




1. Выбирать простую установку. Нажать **Next**.
2. Сетевые параметры можно оставить по умолчанию. Нажать **Next**.

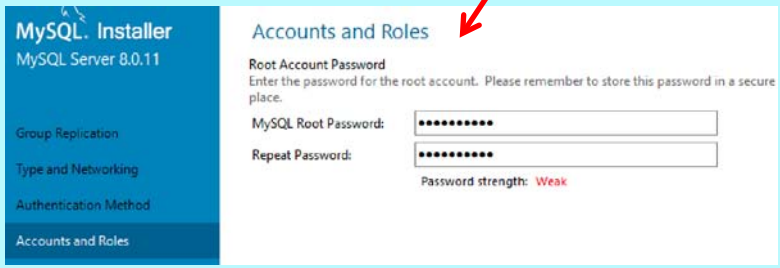
Реляционная СУБД MySQL

Этап настройки



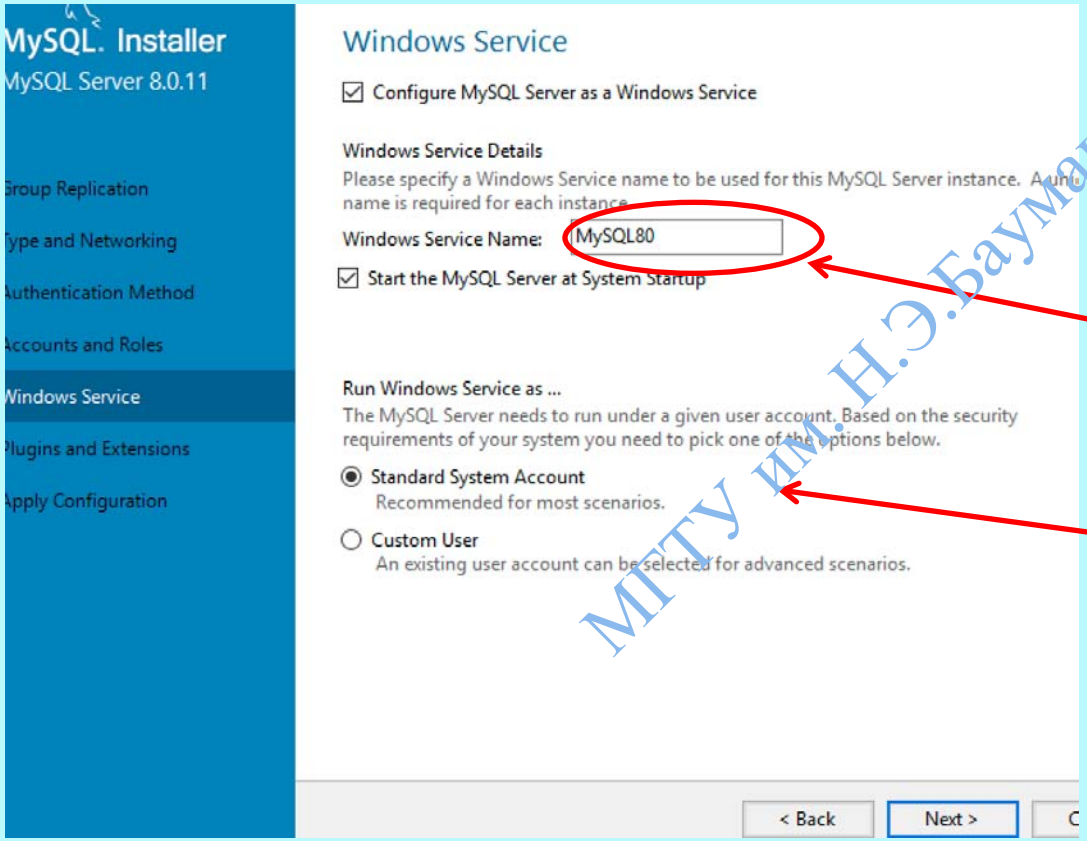
3. Параметры аутентификации. Нажать **Next**.

4. Установить пароль для сервера (и запомнить! Нажать **Next**



Реляционная СУБД MySQL

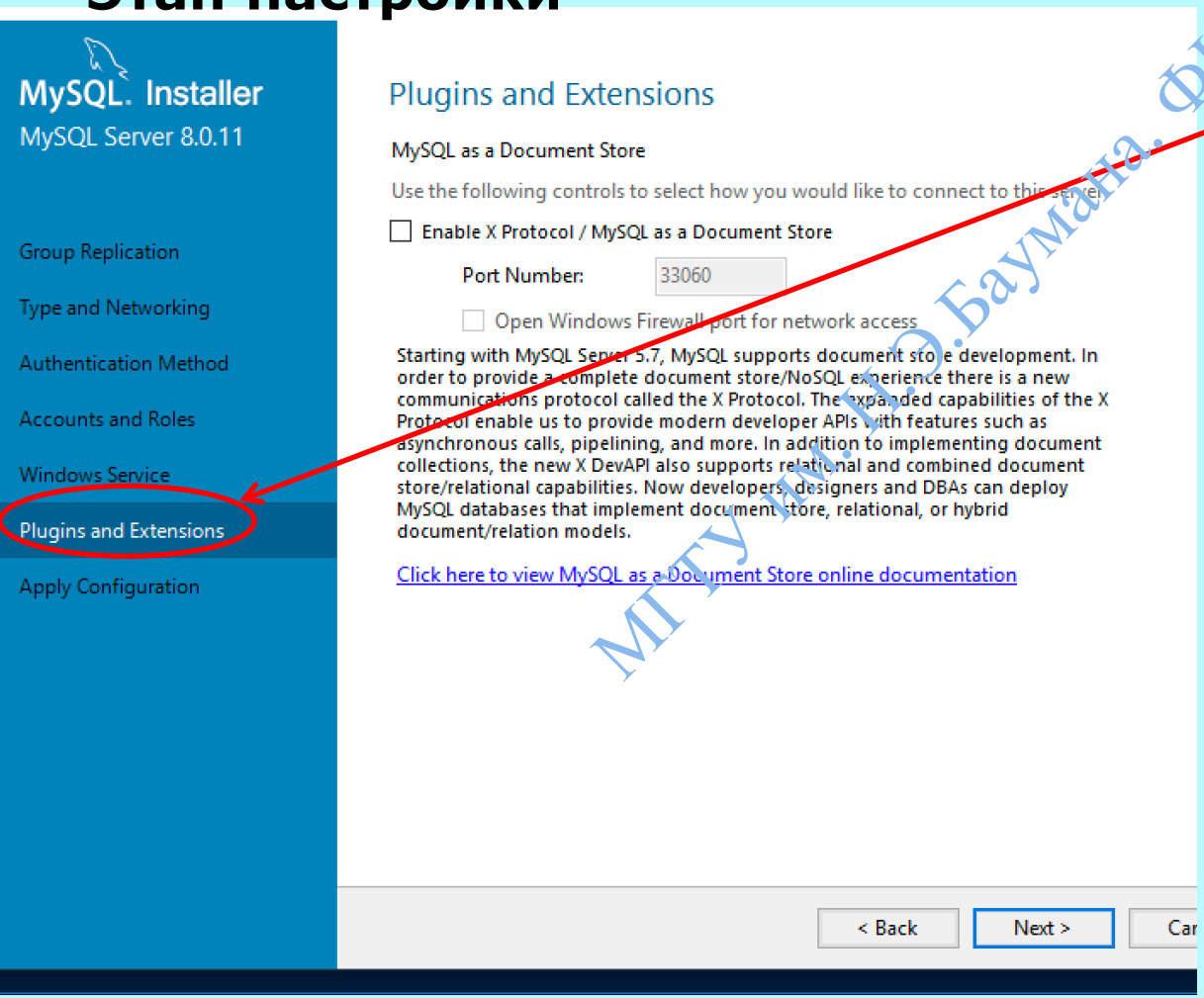
Этап настройки



5. Настроить свойства Window Servis, для этого указать :
Имя Window Servis, параметры автозапуска, учетную запись под которой необходимо запускать данную службу.
Нажать **Next**.

Реляционная СУБД MySQL

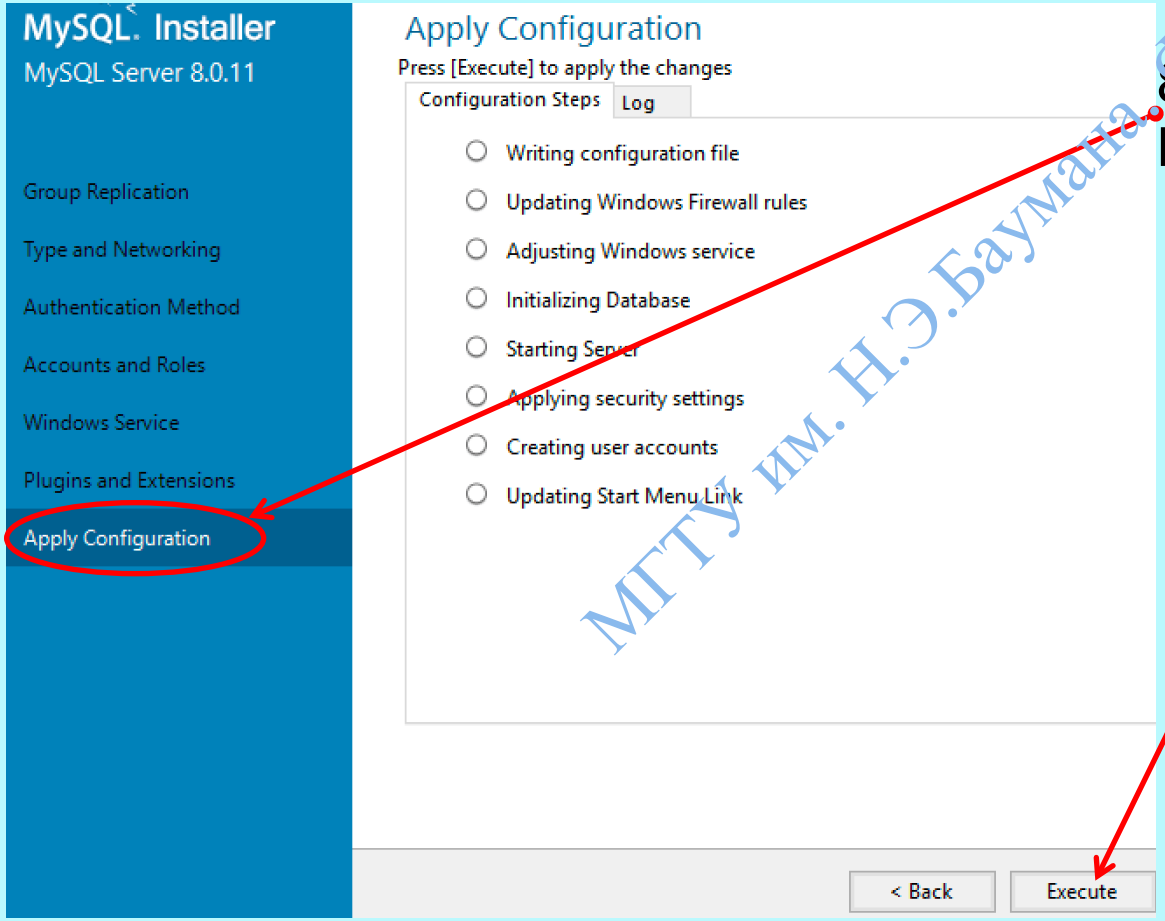
Этап настройки



7. Параметры Plagins and Extensions оставить по умолчанию. Нажать **Next**.

Реляционная СУБД MySQL

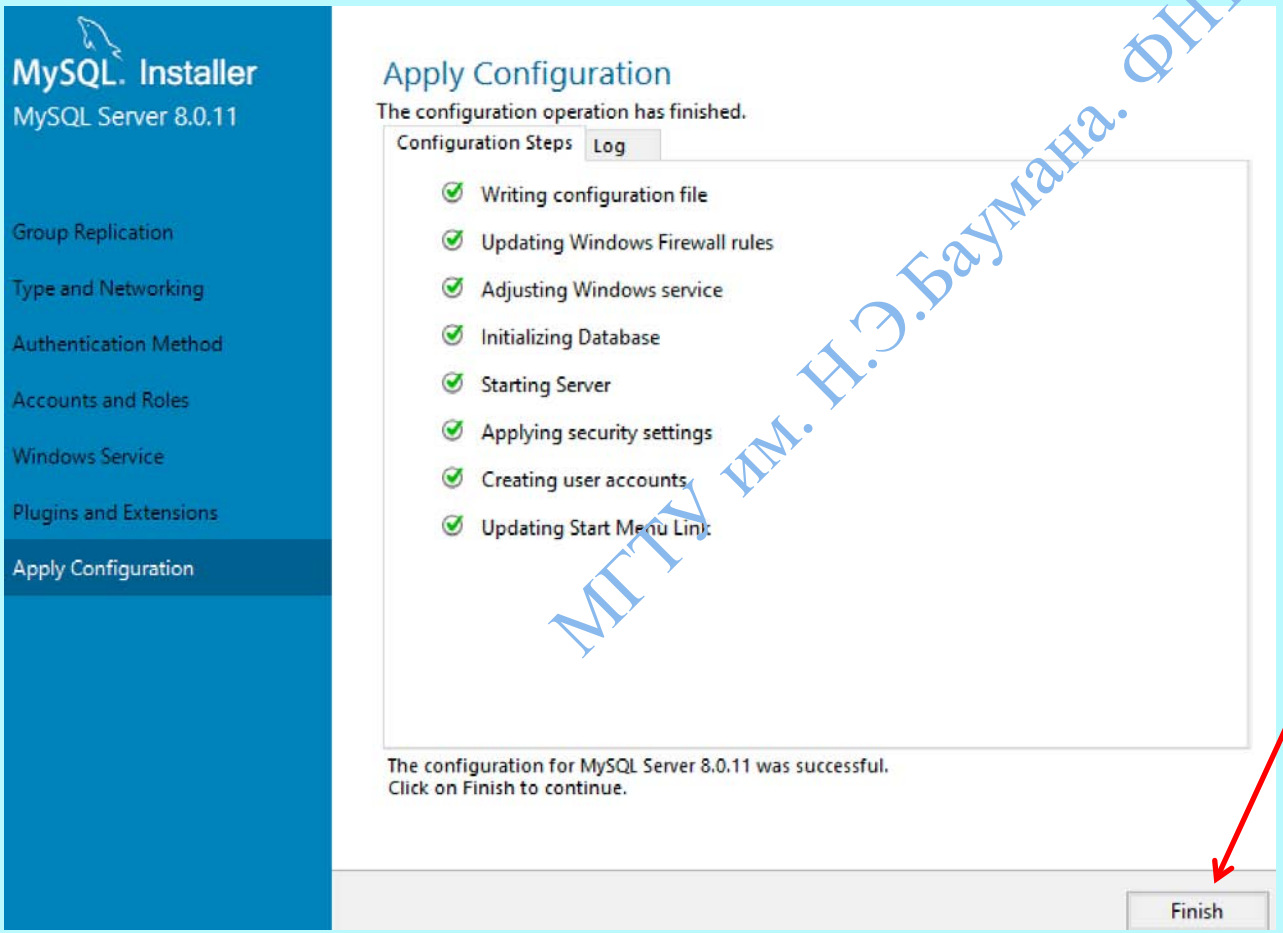
Этап настройки



8. Применить настройки. Нажать **Execute**.

Реляционная СУБД MySQL

Этап настройки



9. Нажать **Finish**.

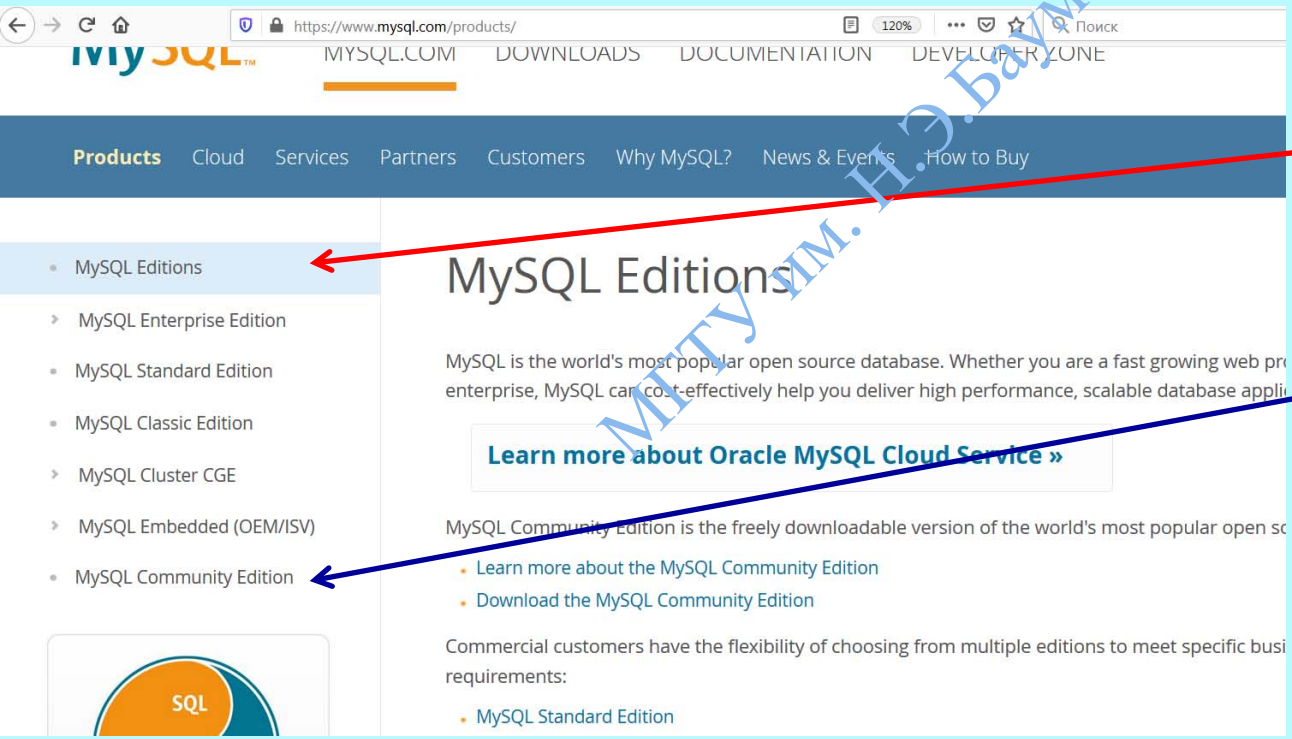
Реляционная СУБД MySQL

Сайт разработчика. Вкладка продукт.

https://www.mysql.com/products/

Установка workbench (в старых версиях продукт был бесплатным)

В случае, если при установке был выбран вариант Server Only (установка только сервера MySQL), можно установить **workbench** или работать с командной строкой.



Здесь выбран вариант **MySQL Edition**

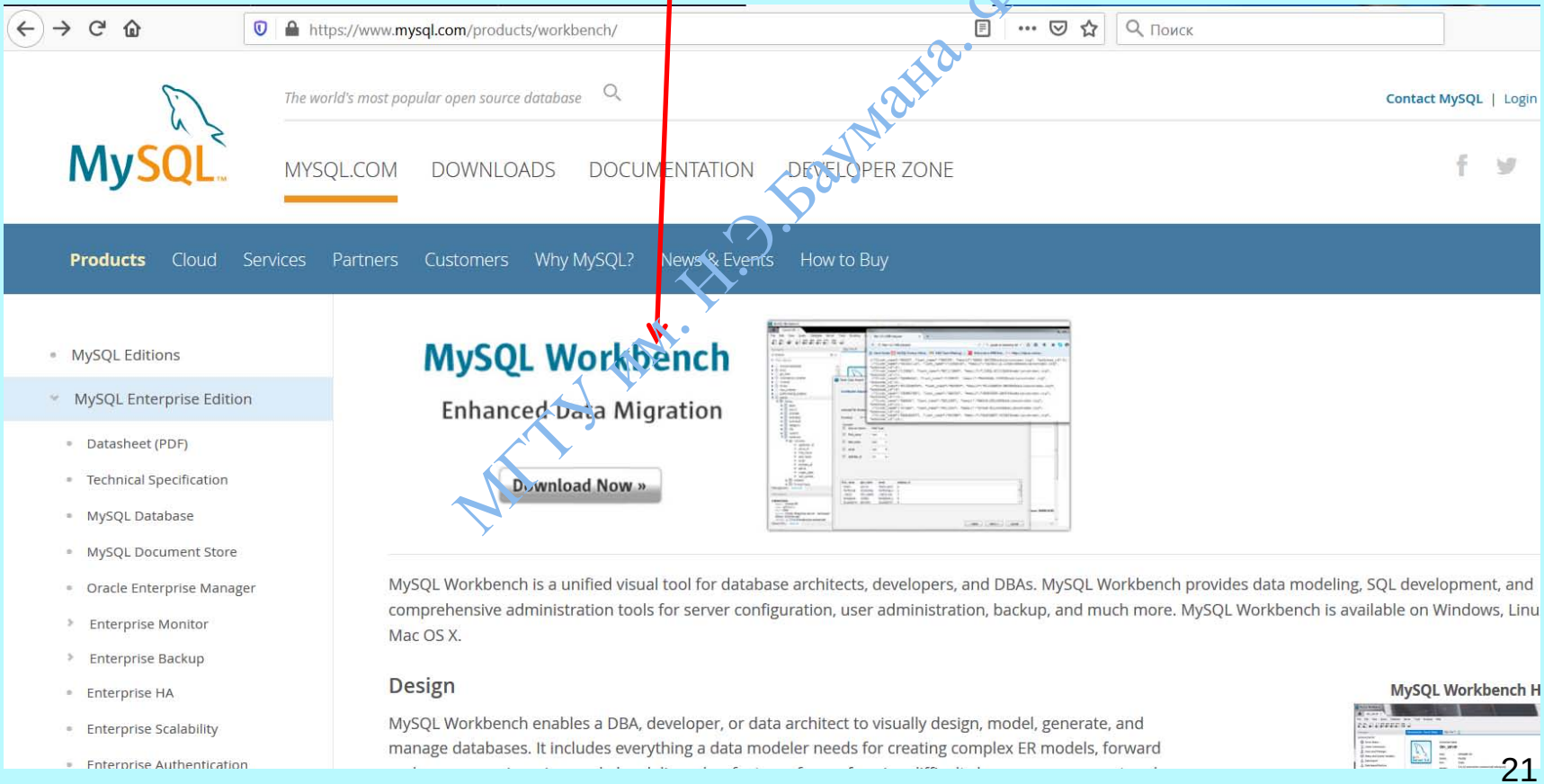
Можно выбрать **Community Edition**

При выборе **Community Edition** устанавливать заранее MySQL Server не требуется

Реляционная СУБД MySQL

https://www.mysql.com/products/workbench/

Скачать **workbench** и далее установить



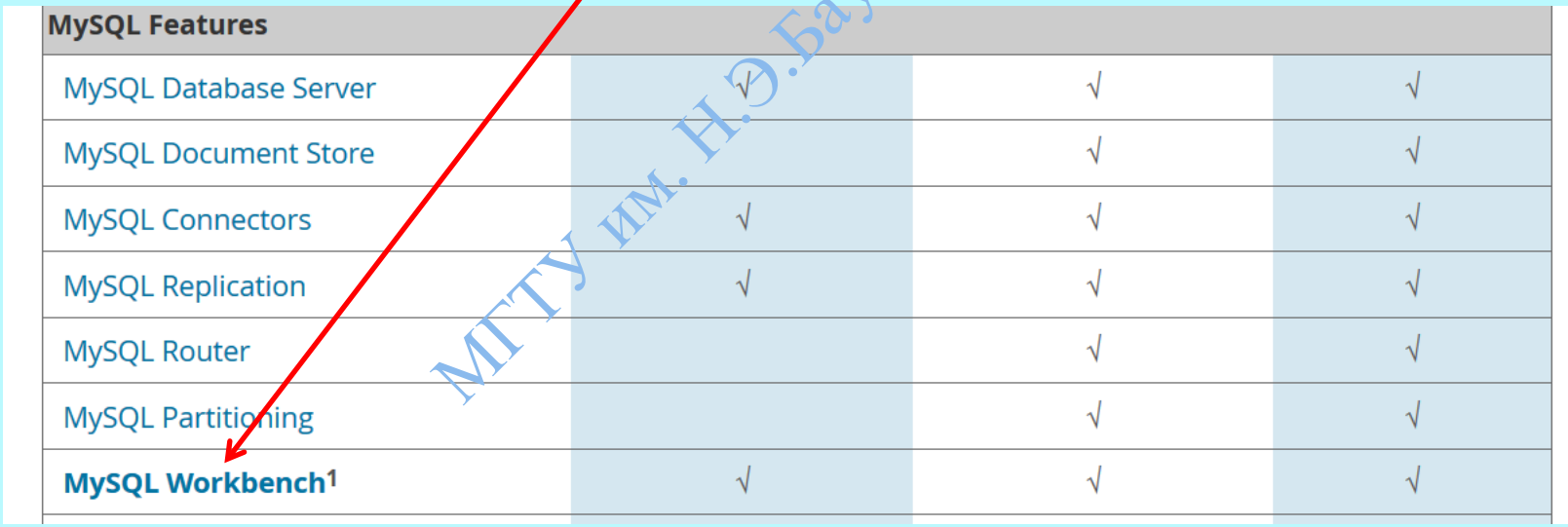
Реляционная СУБД MySQL

Сайт разработчика. Вкладка продукт.

https://www.mysql.com/products/

Установка workbench

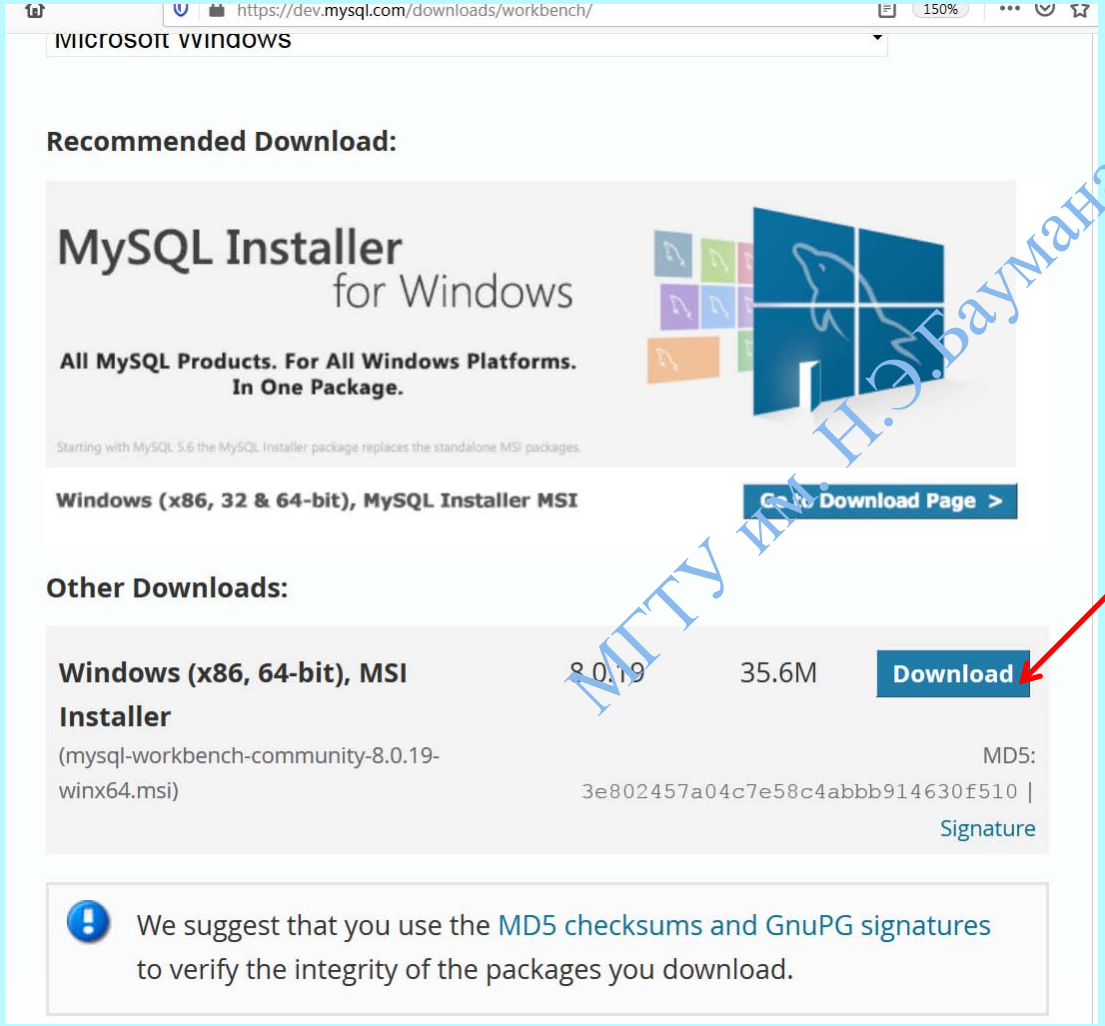
Выбрать вкладку **workbench**



MySQL Features			
MySQL Database Server	√	√	√
MySQL Document Store		√	√
MySQL Connectors	√	√	√
MySQL Replication	√	√	√
MySQL Router		√	√
MySQL Partitioning		√	√
MySQL Workbench[1]	√	√	√

Реляционная СУБД MySQL

https://www.mysql.com/products/workbench/



Выбрать **workbench** и скачать.

Далее установить **workbench**.

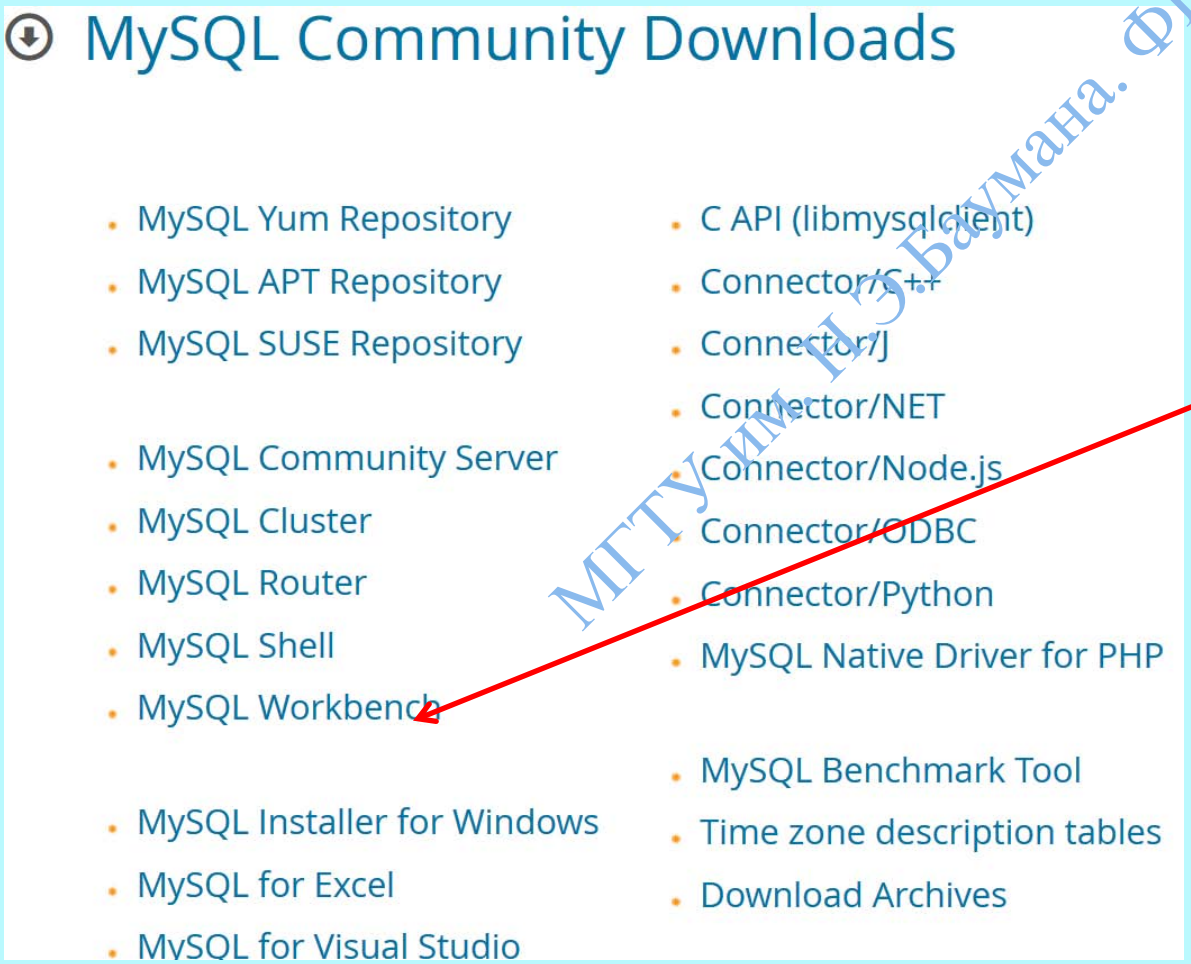
Реляционная СУБД MySQL

Сайт разработчика. Вкладка продукт.

https://www.mysql.com/products/

Community Edition

Установка workbench



Выбрать **workbench**

Реляционная СУБД MySQL

https://www.mysql.com/products/workbench/

Выбрать (скачать без регистрации).
Далее при установке следовать приведенным выше инструкциям по установке MySQL.

ИСТОРИЯ РАЗВИТИЯ СУБД MYSQL

Автор – Michael (Monty) Widenius (TcX, Швеция),

Изначально СУБД разрабатывалась для решения внутренних задач компании (СУБД ISAM, интерфейс в виде экранных форм, не-SQL, конец 80-ых), в 1995 выпущена СУБД MySQL (таблицы ISAM и MyISAM)

Поддержка таблиц InnoDB появилась в MySQL версии 3.23 (1999-2000г.)

2008 – MySQL куплен Sun Microsystems

С 2009 года разработку и поддержку MySQL осуществляет корпорация Oracle.

Widenius разработал ветвь MySQL – MariaDB, принадлежит компании Monty Program Ab.

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.

Характеристика MySQL разработчиками

1. MySQL - это система управления реляционными базами данных
2. Программное обеспечение MySQL - это ПО с открытым кодом
3. Технические возможности СУБД MySQL

ПО MySQL является системой клиент-сервер;

графические клиенты **MySQLQueryBrowser**, PhpMyAdmin и др;

MySQL взаимодействует с базой данных на языке ***SQL*** (***Structured Query Language***).

4. Безопасность
5. Вместимость данных

Начиная с MySQL вер. 3.23 максимальный размер таблицы доведен до 8 миллионов терабайт (2^{63} bytes), размер таблицы в БД MySQL обычно лимитируется операционной системой.

Создание базы данных

I. Работа с командной строкой

1. Запустить сервер MySQL;

2. Вызовать программу клиента mysql, вводя в строке приглашения

mysql;

3. Приглашение изменится на mysql>. Введите команду:

create database Students;

(Примечание: Команда заканчивается символом точки с запятой).

4. Сервер MySQL должен ответит примерно как на рис. 1.1

```
Query OK, 1 rows affected (0.00 sec)
```

Рис. 1.1. Результат работы команды создания таблицы.
(Запрос обработан, изменилась 1 строка (0.00 сек))

5. База данных была успешно создана.

6.Посмотреть какие базы данных имеется в системе можно командой

show databases;

```
Mysql>Show databases;
+-----------------------+
|   Data base           |
+-----------------------+
|  Students             |
|   mysql               |
|   test                |
+-----------------------+
2 row in set (0.00 sec)
```

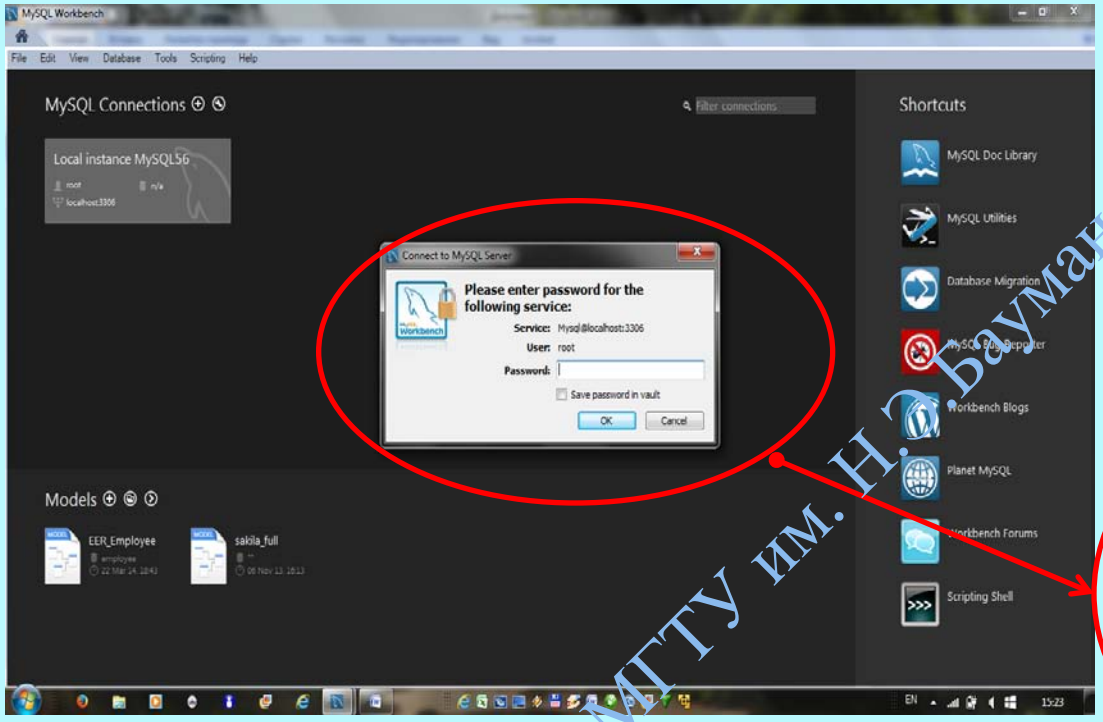
Рис.1.2. Просмотр баз данных

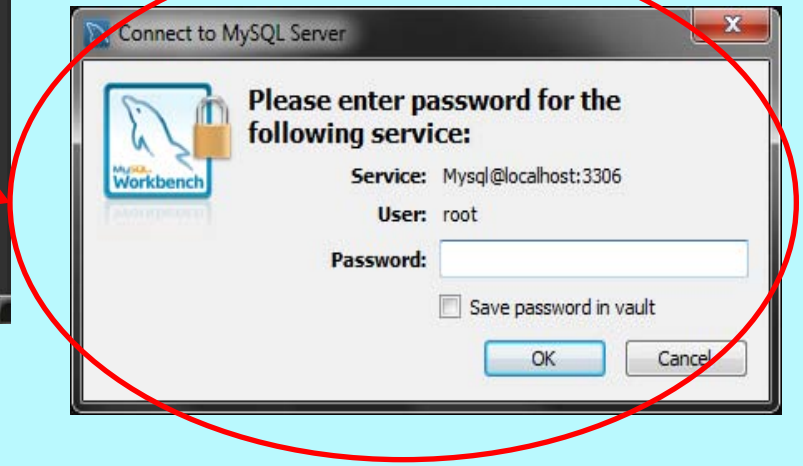
Здесь показаны три базы данных, две были созданы MySQL во время установки и вновь созданная база данных Students.

6. Для завершения работы введите команду quit *mysql> \q*.

II. MySQL Workbench

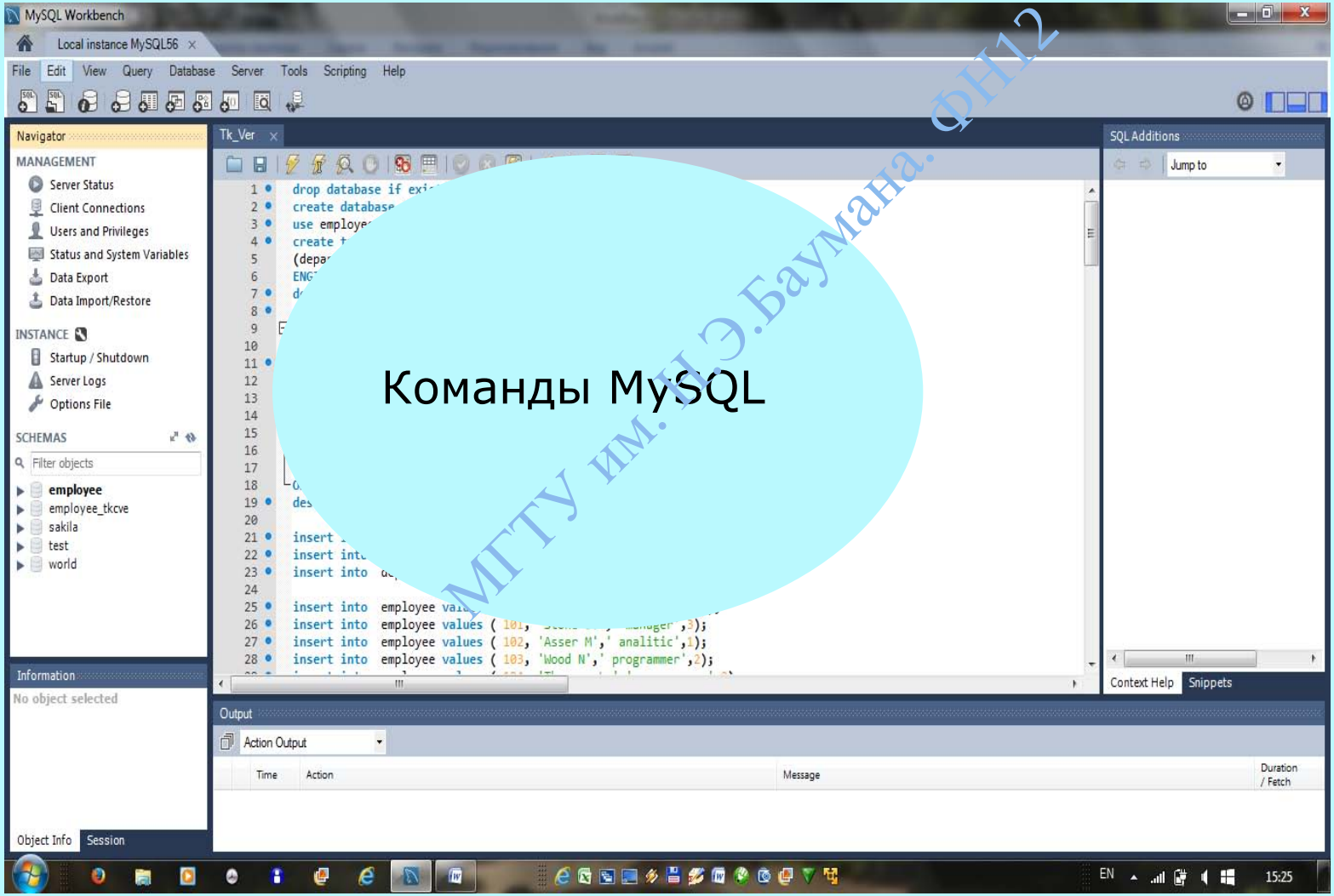
Вызвать **Workbench**. Ввести пароль, заданный при установке MySQL





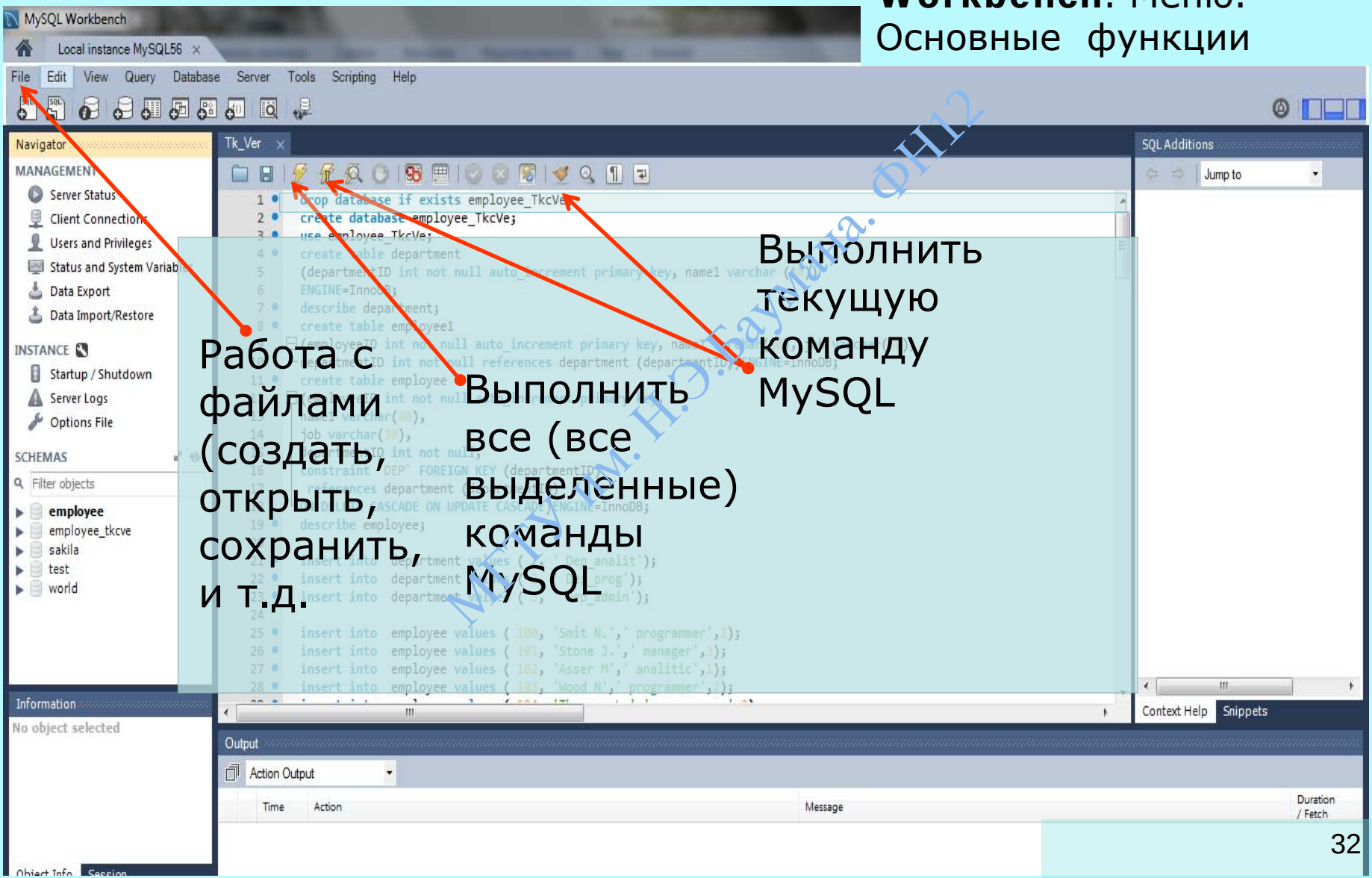
II. MySQL Workbench

Общий вид окна **Workbench**.



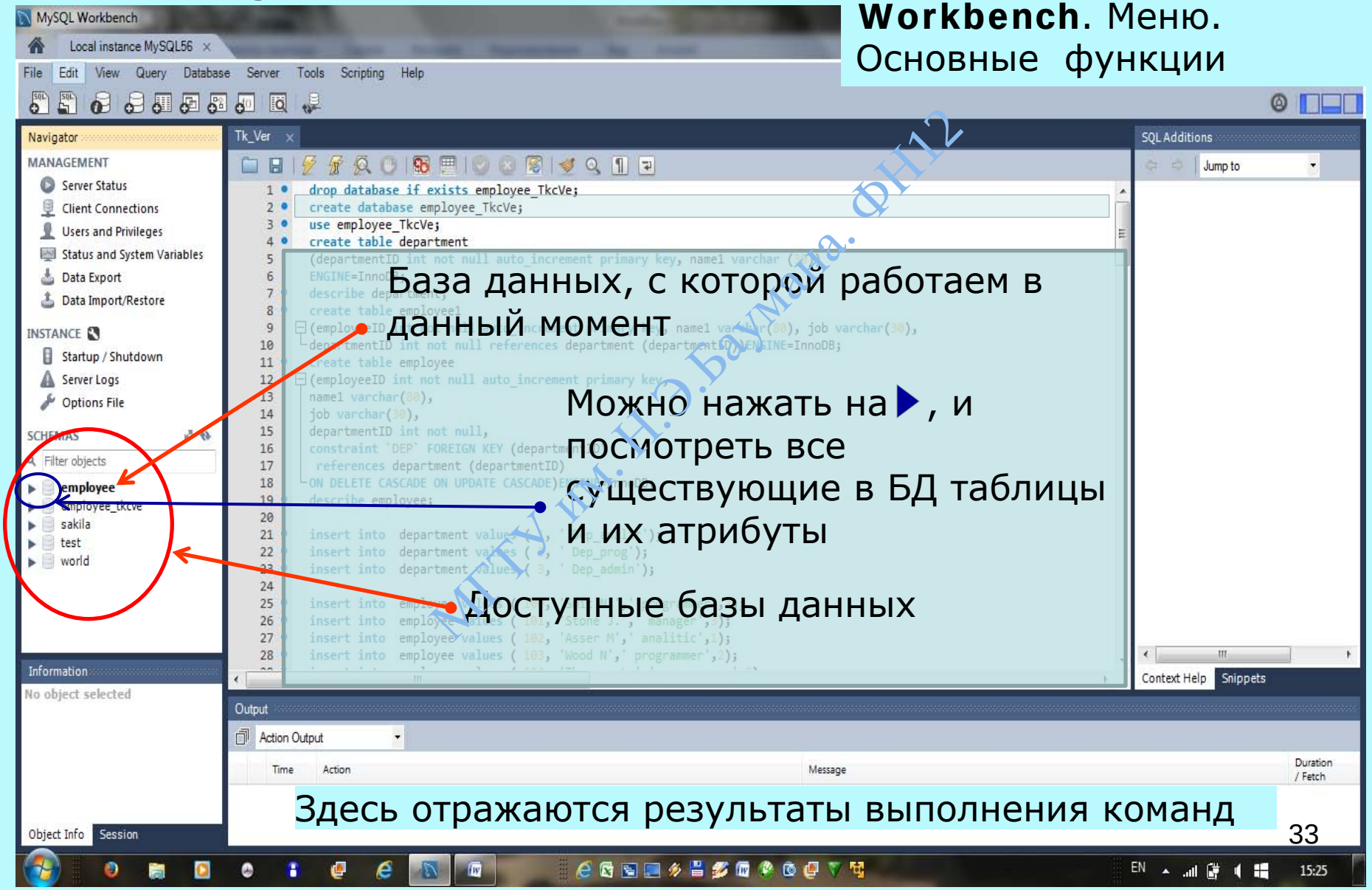
II. MySQL Workbench

Общий вид окна **Workbench**. Меню. Основные функции



II. MySQL Workbench

Общий вид окна **Workbench**. Меню. Основные функции



База данных, с которой работаем в данный момент

Можно нажать на ▶, и посмотреть все существующие в БД таблицы и их атрибуты

Доступные базы данных

Здесь отражаются результаты выполнения команд

Команда CREATE DATABASE

Синтаксис команды :

CREATE DATABASE [IF NOT EXISTS] имя_базы_данных

[спецификация_create[,спецификация_create]...]

Команда **CREATE DATABASE** создает базу данных с указанным именем. Если база данных с таким именем существует, генерируется ошибка.

спецификация_**create**:

[DEFAULT] CHARACTER SET имя_набора_символов [DEFAULT] COLLATE имя_порядка_сопоставления

Опция *спецификация_create* может указываться для определения характеристик базы данных, которые сохраняются в файле db.opt, расположенном в каталоге данных.

Конструкция CHARACTER SET определяет набор символов для базы данных по умолчанию.

Конструкция COLLATION задает порядок сопоставления по умолчанию.

Сопоставление в SQL — это ряд правил, согласно которым сортируются и сравниваются данные.

Выбор БД

При работе необходимо определять базу данных, которая будет использоваться, иначе MySQL будет порождать ошибку.

Определить текущую БД можно

1) при запуске **mysql Students;**
2) с помощью оператора USE в приглашении mysql
   mysql> **USE Students;**
3) с помощью \u в приглашении mysql mysql> **\u Students;**
4) с помощью оператора USE в окне Workbench **USE Students;**

Работа с таблицами

Для *создания таблиц* БД используется команда CREATE TABLE

Синтаксис :

CREATE [TEMPORARY] TABLE [IF NOT EXIST] Имя_Таблицы
[(определение_create,…)] [опции_таблицы] [опции_select];

или

CREATE [TEMPORARY] TABLE [IF NOT EXIST] *Имя_Таблицы*
LIKE Имя_старой_таблицы;

определение_create:

Имя_Столбца **тип** [NOT NULL | NULL] [DEFAULT *значение*]
[AUTO_INCREMENT] [PRIMARY KEY] [REFERENCES] [определение_ссылки]

или PRIMARY KEY (*Имя_Столбца_индекса,..*)

или KEY [*Имя_индекса*] (*Имя_Столбца_индекса,..*)

или INDEX [*Имя_индекса*] (*Имя_Столбца_индекса,..*)

или UNIQUE [INDEX] [*Имя_индекса*] (*Имя_Столбца_индекса,..*)

или FULLTEXT [INDEX] [*Имя_индекса*] (*Имя_Столбца_индекса,..*)

или [CONSTRAINT символ] FOREIGN KEY [имя_индекса]
(*Имя_Столбца_индекса,..*) [определение_ссылки]

или CHECK (выражение)

Значение основных опций

TEMPORARY используется для создания временных таблиц, которые автоматически удаляются после завершения сеанса.

LIKE Имя_старой_таблицы используется для создания таблицы с такой же схемой, как у таблицы *Имя_старой_таблицы*.

В скобках () оператора *CREATE TABLE* объявляются столбцы, их тип и другая информация о структуре таблицы.

Опции в определение столбца :

NOT NULL | NULL столбцу не позволено|позволено содержать значение NULL

DEFAULT объявляет для столбца значение по умолчанию *значение*

AUTO_INCREMENT позволяет автоматически генерировать порядковый номер, увеличивает номер на 1.

PRIMARY KEY позволяет объявить столбец таблицы первичным ключом

REFERENCES позволяет объявить столбец таблицы внешним ключом

Кроме имен столбцов и их типов можно объявить и другую информацию о можно объявить и другую информацию о столбцах.

Чтобы создать первичный ключ, состоящий из *нескольких столбцов* необходимо указать выражение PRIMARY KEY за которым следуют имена столбцов, которые формируют этот ключ. Столбец PRIMARY KEY является уникальным индексированным столбцом, который не может принимать значение NULL.

KEY и INDEX синонимы, означают, что указанные столбцы (столбец) будут индексированы, не обязательно должны содержать уникальные значения.

UNIQUE [INDEX] соответствующий столбец обязательно должен содержать уникальные значения, столбцы (столбец) будут индексированы.

FULLTEXT [INDEX] используется для создания полнотекстовых индексов на основе столбцов типа CHAR, VARCHAR, TEXT. (Только с табл.MyISAM)

FOREIGN KEY позволяет объявить внешние ключи точно так же, как и первичные

Пример 1.

Номер л.д.	ФИО	Группа	Специальность
1232	Alexeev A.A.	Inf12	Inform. System
1233	Borisov B.B.	Inf12	Inform. System
1234	Vasin V.V	Inf12	Inform. System
1235	Grishin G.G	Cnt12	Control. System

```
CREATE DATABASE Students; # создание БВ

USE Students;  # используем БД Students, обязательная команда
CREATE TABLE  Student
(
  StudentID INT NOT NULL AUTO_INCREMENT PRIMARY KEY,
  Name VARCHAR(80),
  Groupe VARCHAR(5),
  Speciality VARCHAR(80)
) ENGINE= InnoDB; # создание отношения (таблицы) Student

  DESCRIBE  Student; # просмотр структуры таблицы
```

Groupe – название (номер) группы. Слово Group является зарезервированным, поэтому называем атрибут Groupe или Groupe 1

Пример 1.

```
CREATE TABLE  Student
(  StudentID INT NOT NULL AUTO_INCREMENT PRIMARY KEY,
  Name VARCHAR(80),   Groupe VARCHAR(5),   Speciality VARCHAR(80)
) ENGINE= InnoDB; # создание отношения (таблицы) Student
```

Атрибут ***StudentID*** имеет целый тип *INT*;
является первичным ключом PRIMARY KEY и поэтому не может принимать значение *NULL*, явно указываем это – NOT NULL.

Тип *AUTO_INCREMENT* говорит о том, что можно не указывать значение этого атрибута при вводе данных в таблицу.
По умолчанию значение равняется 1 и с каждой новой записью увеличивается на 1. В случае явного задания значения атрибута *StudentID* в БД будет храниться это значение.
Если значения атрибута явно не задано, то в новое значение атрибута *StudentID* будет на 1 больше последнего записанного в БД.
В случае удаления каких либо записей(не с наибольшим значением *AUTO_INCREMENT*), значения атрибутов типа *AUTO_INCREMENT* не изменяются, новое значение все равно будет на 1 больше максимального значения данного поля.
Значение *AUTO_INCREMENT* будет использовано повторно, если удаляется строка, содержащая наибольшее значение *AUTO_INCREMENT*.

Пример 1.

```
CREATE TABLE  Student
(  StudentID INT NOT NULL AUTO_INCREMENT PRIMARY KEY,
  Name VARCHAR(80),   Groupe VARCHAR(5),   Speciality VARCHAR(80)
) ENGINE= InnoDB; # создание отношения (таблицы) Student
```

Атрибуты *Name* и *Groupe* имеют тип *VARCHAR(n)*, это означает, что данные атрибуты имеют строковый (символьный) тип, и могут иметь не более *n* символов.

ENGINE= InnoDB.
означает, что таблица *Student является транзакционной таблицей* типа InnoDB.

InnoDB является системой управления базой данных в рамках MySQL.
В InnoDB есть свой собственный буферный пул для кэширования данных и индексов в основной памяти.
Таблицы и индексы InnoDB хранятся в специальном пространстве памяти, которое может состоять из нескольких файлов, которые могут находится на нескольких дисках или хостах (каждая таблица MyISAM хранится в отдельном файле).
Таблицы InnoDB могут быть практически любого размера .

Типы данных

Целые

TINYINT[(M)] [UNSIGNED] [ZEROFILL]	Очень малое целое число. Диапазон: со знаком [-128 - 127],без знака [0 – 255] Число отображается с ведущими нулями
BIT, BOOL	Синонимы TINYINT(1)
SMALLINT[(M)] [UNSIGNED] [ZEROFILL]	Малое целое число. Диапазон: со знаком [-32 768 – 32 767], без знака [0 – 65 535].
INT[(M)] [INTEGER] [UNSIGNED[(M)]] [ZEROFILL]	Целое число нормального размера. Диапазон: со знаком [-2 147 483 648 – 2 147 483 647], без знака [0 – 4 294 967 295].
MEDIUMINT[(M)] [UNSIGNED] [ZEROFILL]	Целое число среднего размера. Диапазон: со знаком [-8 388 608 – 8 388 607], без знака [0 – 16 777 215].
BIGINT[(M)] [UNSIGNED] [ZEROFILL]	Большое целое число. Диапазон со знаком [-9 223 372 036 854 775 808 – 9 223 372 036 854 775 807], без знака [0 – 18 446 744 073 709 551 615]

Вещественные

FLOAT[(M,D)] [UNSIGNED] [ZEROFILL]	Малое число с плавающей точкой обычной точности. Диапазон: [-3,40282E+38 ÷ -1,17549E -38], 0, [1,17549E -38 ÷ 3,40282E+38]. Атрибут UNSIGNED указывает, что отрицательные значения недопустимы. Атрибут M указывает количество выводимых пользователю знаков. Атрибут D указывает количество разрядов, следующих за десятичной точкой.
FLOAT(Pres) [UNSIGNED] [ZEROFILL]	Число с плавающей точкой. Атрибут Pres точность имеет значение : Pres≤24 для числа с обычной (одинарной) точности; 25 ≤ Pres ≤ 53 - для числа с удвоенной точности. Эти типы данных сходны с типами FLOAT и DOUBLE,. FLOAT(X) относится к тому же интервалу, что и соответствующие типы FLOAT и DOUBLE, но диапазон значений и количество десятичных знаков не определены.

DOUBLE[(M,D)] [UNSIGNED] [ZEROFILL]	Число с плавающей точкой удвоенной точности нормального размера. Диапазон: [-1,79769E+308 ÷ -2,22507E-308], 0, [2,22507E -308 ÷ 1,79769E+308]. Атрибут UNSIGNED указывает, что отрицательные значения недопустимы. Атрибут M указывает количество выводимых пользователю знаков. Атрибут D указывает количество разрядов, следующих за десятичной точкой.
DECIMAL[(M[,D])] [UNSIGNED] [ZEROFILL] или DEC[(M[,D])] [UNSIGNED] [ZEROFILL] или NUMERIC[(M[,D])] [UNSIGNED] [ZEROFILL]	"Неупакованное" число с плавающей точкой. Подобно столбцу CHAR, содержащему цифровое значение. Разделительный знак десятичных разрядов, знак '-' для отрицательных чисел не учитываются в M (место для них зарезервировано). При D=0, величины будут представлены без дробной части. Максимальный интервал значений типа DECIMAL тот же, что и для типа DOUBLE, но действительный интервал для конкретного столбца DECIMAL может быть ограничен выбором значений атрибутов M и D.. По умолчанию: значение D = 0. Значение M = 10.

Дата- время

DATE	Дата. Поддерживается интервал от '1000-01-01' до '9999-12-31'. MySQL выводит значения DATE в формате 'YYYY-MM-DD', но можно установить значения в столбец DATE, используя как строки, так и числа.
DATETIME	Комбинация даты и времени. Поддерживается интервал от '1000-01-01 00:00:00' до '9999-12-31 23:59:59'. MySQL выводит значения DATETIME в формате 'YYYY-MM-DD HH:MM:SS', но можно устанавливать значения в столбце DATETIME, используя как строки, так и числа.
TIMESTAMP[(M)]	Временная метка. Интервал от '1970-01-01 00:00:00' до некоторого значения времени в 2037 году. MySQL выводит значения TIMESTAMP в форматах YYYYMMDDHHMMSS, YYMMDDHHMMSS, YYYYMMDD или YYMMDD в зависимости от значений M: 14 (или отсутствующее), 12, 8, или 6; но можно также устанавливать значения, используя как строки, так и числа.
TIME	Время. Интервал от '-838:59:59' до '838:59:59'. MySQL выводит значения TIME в формате 'HH:MM:SS', но можно устанавливать значения в столбце TIME, используя как строки, так и числа.

YEAR[(2|4)]	Год в двухзначном или четырехзначном форматах (по умолчанию формат четырехзначный). Допустимы следующие значения: с 1901 по 2155, 0000 для четырехзначного формата года и 1970-2069 при использовании двухзначного формата (70-69). MySQL выводит значения YEAR в формате YYYY, но можно задавать значения в столбце YEAR, используя как строки, так и числа.
[NATIONAL] CHAR(M) [BINARY]	Строка фиксированной длины, при хранении всегда дополняется пробелами в конце строки до заданного размера. Диапазон аргумента M составляет от 0 до 255 символов. Концевые пробелы удаляются при выводе значения. Если не задан атрибут чувствительности к регистру BINARY, то величины CHAR сортируются и сравниваются как независимые от регистра в соответствии с установленным по умолчанию алфавитом. Атрибут NATIONAL CHAR (или его эквивалентная краткая форма NCHAR) представляет собой принятый в ANSI SQL способ указания, что в столбце CHAR должен использоваться установленный по умолчанию набор символов (CHARACTER).
CHAR	Это синоним для CHAR(1).

Строки

[NATIONAL] VARCHAR(M) [BINARY]	Строка переменной длины. Примечание: концевые пробелы удаляются при сохранении значения (в этом заключается отличие от спецификации ANSI SQL). Диапазон аргумента M составляет от 0 до 255 символов. Если не задан атрибут чувствительности к регистру BINARY, то величины VARCHAR сортируются и сравниваются как независимые от регистра
TINYBLOB, TINYTEXT	Столбец типа BLOB или TEXT с максимальной длиной 255 символов
BLOB, TEXT	Столбец типа BLOB или TEXT с максимальной длиной 65535 символов.
MEDIUMBLOB, MEDIUMTEXT	Столбец типа BLOB или TEXT с максимальной длиной 16777215 символов.
LONGBLOB, LONGTEXT	Столбец типа BLOB или TEXT с максимальной длиной 4294967295 символов.

Перечисляемый тип данных и тип данных множество

ENUM('значение1','значение2',...)	Перечисляемый тип данных. Объект строки может иметь только одно значение, выбранное из заданного списка величин 'значение1', 'значение2', ..., NULL или специальная величина ошибки "". Список ENUM может содержать максимум 65535 различных величин
SET('значение1','значение2',...)	Набор. Объект строки может иметь ноль или более значений, каждое из которых должно быть выбрано из заданного списка величин 'значение1', 'значение2', ... Список SET может содержать максимум 64 элемента.

Удаление таблиц

Для удаления таблицы используется команда ***DROP TABLE***.

Синтаксис команды DROP TABLE

DROP TABLE [IF EXISTS] таблица [RESTRICT | CASCADE]

Спецификация IF EXISTS подавляет вывод сообщения об ошибке, выдаваемого в случае, если заданная таблица не существует. Можно указывать несколько имен таблиц, разделяя их запятыми. Необязательные параметры *[RESTRICT | CASCADE]* указываются при наличии **внешних ключей** .

Удалить можно только существующую таблицу.

Проверить существование таблицы можно с помощью команды *SHOW TABLES*

```
Mysql>SHOW TABLES;
+----------------------------+
| Tables in Students         |
+----------------------------+
|Student                     |
+----------------------------+
1 row in set (0.00 sec)
```

Рис. 1.7. Просмотр таблиц в базе

```
Mysql> DROP TABLE Student;
Query OK, 0 rows affected (0.01 sec)
```

Рис. 1.8. Удаление таблицы

Вставка, обновление и удаление данных в MySQL

Использование оператора INSERT.

Оператор **INSERT** заполняет таблицу данными.

Общая форма INSERT.

INSERT [LOW_PRIORITY | DELAYED] [IGNORE] [INTO] *имя_таблицы* [(*имя_столбца*, ...)]VALUES((*выражение* | DEFAULT), ...)

[ON DUPLICATE KEY UPDATE *имя_столбца=выражение*, …];

Или

INSERT [LOW_PRIORITY | DELAYED] [IGNORE] [INTO] *имя_таблицы* [(*имя_столбца*, ...)] SELECT …;

(выбрать строки из другой таблицы)

Или

INSERT [LOW_PRIORITY | DELAYED] [IGNORE] [INTO] *имя_таблицы* SET *имя_столбца*=(*выражение* | DEFAULT), ...

[ON DUPLICATE KEY UPDATE *имя_столбца=выражение*, …];

Задание 1.

1. Создать БД Student. (см. **Пример 1**)
2. Создать таблицу Student.
3. Вставить данные в таблицу.

Номер л.д.	ФИО	Группа	Специальность
1232	Alexeev A.A.	Inf12	Inform. System
1233	Borisov B.B.	Inf12	Inform. System
1234	Vasin V.V	Inf12	Inform. System
1235	Grishin G.G	Cnt12	Control. System

Информация для заполнения таблицы Student (CHARACTER SET -Latin)

INSERT INTO Student (*StudentID* , *Name*, *Groupe*, speciality)
VALUES (1232,'Alexeev A.A.', 'Inf12', 'Inform. System'),
(1233,'Borisov B.B.', 'Inf12', 'Inform. System');

Задание 1.

INSERT INTO Student (*StudentID* , *Name*, *Groupe*, speciality)
VALUES (1232,'Alexeev A.A.', 'Inf12', 'Inform. System');

Номер 1232 задан как начало отсчета для типа AUTO_INCREMENT.
Если вставляем в БД весь кортеж целиком, порядок записи значений атрибутов соответствует порядку атрибутов в операторе CREATE TABLE, то можно не перечислять имена атрибутов в команде INSERT

INSERT INTO Student VALUES (1232,'Alexeev A.A.', 'Inf12', 'Inform. System');

При использовании команды INSERT вставлять в БД можно один или несколько кортежей сразу.

INSERT INTO Student (*Name*, *Groupe*, speciality) VALUES (
'Borisov B.B.', 'Inf12', 'Inform. System'1232),
('Vasin V.V.', 'Inf12', 'Inform. System'),('Grishin G.G.', 'Cnt12', '
Control. System');

Здесь перечисление имен атрибутов обязательно, так как в списке значений атрибутов нет значения атрибута StudentID

Использование оператора REPLACE

REPLACE [LOW_PRIORITY | DELAYED] [IGNORE] [INTO]
имя_таблицы [(*имя_столбца*, ...)]VAlLUES(*выражение*, ...);
Или
REPLACE [LOW_PRIORITY | DELAYED] [IGNORE] [INTO]
имя_таблицы [(*имя_столбца*, ...)]SELECT …;
Или
REPLACE [LOW_PRIORITY | DELAYED] [IGNORE] [INTO]
имя_таблицы SET *имя_столбца=выражение*,
имя_столбца=выражение,...
[ON DUPLICATE KEY UPDATE *имя_столбца=выражение*, …];

REPLACE INTO Student (*StudentID* , *Name*, *Groupe*, speciality
) VALUES (1232,'Alexeev A.A.', 'Inf-12', 'Inform. System'),
(1233,'Borisov B.B.', 'Inf-12', 'Control. System');

Использование оператора DELETE

DELETE [LOW_PRIORITY] [QUICK] FROM *имя_таблицы* [WHERE определение _*where*][ORDER BY…][LIMIT строки];

Или

DELETE [LOW_PRIORITY] [QUICK] *имя_таблицы*[.*][*имя_таблицы*[.*]…] FROM *имя_таблицы*-ссылки [WHERE определение _*where*];

Или

DELETE [LOW_PRIORITY] [QUICK] FROM *имя_таблицы*[.*] [*имя_таблицы* [.*]…]USING *имя_таблицы*-ссылки[WHERE определение _*where*];

```
DELETE FROM Student WHERE StudentID=1232;
```

```
REPLACE INTO Student
(StudentID , Name,  Groupe,  speciality ) VALUES
(1232,‘Alekseev A.A.‘,  ‘Inf-12‘, ’Inform. System’);
```

Использование оператора TRUNCATE

TRUNCATE TABLE *имя_таблицы;*Запрос удалит все записи из таблицы.

Использование оператора UPDATE

UPDATE [LOW_ PRIORITY] [IGNORE] *имя_таблицы* SET
имя_столбца1=выражение1 [, имя_столбца2=выражение2 …]
[WHERE *определение _where*]
[ORDER BY…][LIMIT строки];
Или
UPDATE [LOW_ PRIORITY] [IGNORE] *имя_таблицы*
[,*имя_таблицы …*]
SET *имя_столбца1=выражение1*
[, *имя_столбца2=выражение2*] [WHERE *определение _where*];

Заполнение таблицы данными из файла

Команда **LOAD DATA INFILE** позволяет вставлять данные из текстового файла в одну таблицу. В файле поля записи (значения столбцов) разделяются знаками табуляции, каждая запись (строка данных) размещается в списке в отдельной строке.

*Конец строки в PC- подобных машинах (Window) - **CR-LF .***

*В UNIX -**LF**. Например, в редакторе Aditor при сохранении файла надо указать **CR-Lf type UNIX***

Файл должен находится в каталоге
C:\Program Files\MySQL\MySQL Server 4.1\data\,
иначе надо прописывать полный путь (d:\\Dir1\\ ..\\FileName).

LOAD DATA INFILE:

LOAD DATA [LOW_ PRIORITY | CONCURRENT] [LOCAL]
INFILE ‘имя_файла.txt’ [REPLACE | IGNORE]INTO TABLE
имя_таблицы
[FIELDS
[TERMINANATED BY ‘\t’] *(‘\t’ табуляция)*
[[OPTIONALLY] ENCLOSED BY ‘ ‘]
ESCAPED BY ‘\\’]
[LINES TERMINANATED BY ‘\n’] *(‘\ n’ перевод строки LF)*
[IGNORE число LINES]
[(*имя_столбца, …*)];

Опции:
LOW_ PRIORITY | CONCURRENT-заставляет ждать других клиентов | позволяет работать другим клиентам.

REPLACE | IGNORE – заменять | не заменять старую строку новой.

FIELDS, LINES- сообщают, как размещены данные в файле.

IGNORE число LINES- игнорировать число первых строк в файле.

(TERMINANATED BY ‘ch’ –поля (строки) разделяются символом ch).

имя_столбца- заставляют читать данные только из некоторых столбцов таблицы

Заполнение таблицы данными из файла

Заполнение таблицы STUDENT данными с помощью файла *student.txt*.

Формат команды

LOAD DATA INFILE ‘student.txt’ INTO TABLE *STUDENT*;

Пример файла

NULL Иванов А. Б И-11 ИС 10

NULL Петров А. Б И-11 ИС 10

Измененение структуры таблицы.

Синтаксис оператора *ALTER TABLE*

ALTER [IGNORE] TABLE tbl_name alter_spec [, *alter_spec* ...]

***alter_specification**:*

- ADD [COLUMN] create_definition [FIRST | AFTER column_name] или
- ADD [COLUMN] (create_definition, create_definition,...) или
- ADD INDEX [index_name] (index_col_name,...) или
- ADD PRIMARY KEY (index_col_name,...) или
- ADD UNIQUE [index_name] (index_col_name,...) или
- ADD FULLTEXT [index_name] (index_col_name,...) или
- ADD [CONSTRAINT symbol] FOREIGN KEY index_name (index_col_name,...) [reference_definition] или

- ALTER [COLUMN] col_name {SET DEFAULT literal | DROP DEFAULT} или
- CHANGE [COLUMN] old_col_name create_definition [FIRST | AFTER column_name] или
- MODIFY [COLUMN] create_definition [FIRST | AFTER column_name] или
- DROP [COLUMN] col_name или
- DROP PRIMARY KEY или
- DROP INDEX index_name или
- DISABLE KEYS или
- ENABLE KEYS или
- RENAME [TO] new_tbl_name или
- ORDER BY col или
- table_options

Оператор ALTER TABLE обеспечивает возможность изменять структуру существующей таблицы. Например, можно добавлять или удалять столбцы, создавать или уничтожать индексы или переименовывать столбцы либо саму таблицу. Можно также изменять комментарий для таблицы и ее тип.

Отметим, что при использовании любой другой опции для ALTER TABLE кроме RENAME, MySQL всегда будет создавать временную таблицу, даже если данные, строго говоря, и не нуждаются в копировании

Создать таблицу t1 :

```
CREATE TABLE t1 (pk1 INTEGER NOT NULL AUTO_INCREMENT

PRIMARY KEY, a INTEGER, b CHAR(10));
```

Для того чтобы переименовать таблицу из t1 в t2:

```
ALTER TABLE t1 RENAME t2;
```

Для того чтобы изменить тип столбца с INTEGER на TINYINT NOT NULL (оставляя имя прежним) и изменить тип столбца b с CHAR(10) на CHAR(20) с переименованием его с b на c:

```
ALTER TABLE t2 MODIFY a TINYINT NOT NULL, CHANGE b c
CHAR(20);
```

Для того чтобы добавить новый столбец TIMESTAMP с именем d:

```
mysql> ALTER TABLE t2 ADD d TIMESTAMP;
```

Для того чтобы добавить индекс к столбцу d и сделать столбец a первичным ключом:

```
mysql> ALTER TABLE t2 ADD INDEX (d), ADD PRIMARY KEY (a);
```

Для того чтобы удалить столбец c:

```
mysql> ALTER TABLE t2 DROP COLUMN c;
```

Для того чтобы добавить новый числовой столбец AUTO_INCREMENT с именем c:

```
mysql> ALTER TABLE t2 ADD c INT UNSIGNED NOT NULL
AUTO_INCREMENT
```

Для того чтобы удалить первичный ключ:

```
mysql>  ALTER TABLE t2 DROP PRIMARY KEY
```

Замечание.
1. Таблица должна быть пустой.
2. Если первичный ключ имеет атрибут AUTO_INCREMENT, то сначала нужно снять этот атрибут с индексного поля, следующей командой:

```
mysql> ALTER TABLE t2 CHANGE pk1 pk1 INTEGER NOT
NULL;
```

Затем ввести команду

```
mysql>  ALTER TABLE t2 DROP PRIMARY KEY;
```

Вернем первичный ключ обратно

```
mysql>  ALTER TABLE t2 CHANGE pk1 pk1 INT NOT
NULL AUTO_INCREMENT PRIMARY KEY;
```

Пример 2.

```
USE Students;

DESCRIBE  Student;
```

Добавим столбец «пропуски занятий»

```
ALTER TABLE Student ADD absence INT;

DESCRIBE  Student;

UPDATE Student  SET absence =1  WHERE StudentID=1234;
```

Запрос данных из таблицы MySQL

Оператор ***SELECT*** имеет следующий формат:

SELECT имена_столбцов from имя_таблицы [WHERE ...условия];

Чтобы узнать фамилии всех студентов, необходимо выполнить следующую команду.

SELECT *Name* from *Student*;

Чтобы вывести всю таблицу, можно либо ввести имена всех столбцов, либо воспользо-ваться упрощенной формой оператора SELECT.

SELECT * from *Student*;

Выборка данных с помощью условий

Теперь более подробно рассмотрим формат оператора **SELECT**.

Формат оператора **SELECT** имеет вид:

SELECT *имена_столбцов* FROM *имя_таблицы*
[WHERE *...условия*]
[GROUP BY *группа*[HAVING *групповые_условия*]]
[ORDER BY сортировка_столбцов]
[LIMIT пределы];

Оператор SELECT без условий выводит все данные из указанных столбцов.

Выборка данных с помощью условий

Полный формат оператора **SELECT** имеет вид:

```
SELECT [ ALL | DISTINCT | DISTINCTROW ]
[ HIGH_PRIORITY ]
[ STRAIGHT_JOIN ]
[ SQL_SMALL_RESULT | SQL_BIG_RESULT ] [ SQL_BUFFER_RESULT ]
[ SQL_CACHE | SQL_NO_CACHE ]
[ SQL_CALC_FOUND_ROWS ]
expressions  FROM tables
[WHERE conditions]
[GROUP BY expressions]
[HAVING condition]
[ORDER BY expression [ ASC | DESC ]]
[LIMIT [offset_value] number_rows | LIMIT number_rows OFFSET
offset_value]
[PROCEDURE proc_nm]
[INTO [ OUTFILE 'file_name' options
| DUMPFILE 'file_name'
| @variable1, @variable2, … @variable_n]
[FOR UPDATE | LOCK IN SHARE MODE];
```

Выборка данных с помощью условий

Параметры SELECT

ALL — необязательный параметр. Возвращает все совпадающие строки

DISTINCTROW или DISTINCT— необязательный параметр. Удаляет дубликаты из набора результатов.

HIGH_PRIORITY — необязательный параметр, оператор запускает SELECT перед любыми операторами UPDATE, ожидающими того же ресурса, может использоваться с таблицами MyISAM, MEMORY и MERGE, которые используют блокировку на уровне таблицы.

STRAIGHT_JOIN — необязательный параметр. Соединение таблиц происходит в том порядке, в котором они перечислены в предложении FROM.

SQL_SMALL_RESULT — необязательный параметр. Использует быстрые временные таблицы для хранения результатов (используется с DISTINCT и GROUP BY).

SQL_BIG_RESULT — необязательный параметр. Используется сортировка, а не временная таблица для хранения результатов (используется с DISTINCT и GROUP BY).

SQL_BUFFER_RESULT — необязательный параметр. Используются временные таблицы для хранения результатов (не используется с подзапросами).

Выборка данных с помощью условий

Параметры SELECT

SQL_CACHE — необязательный параметр. Сохраняет результаты в кеше запросов.
SQL_NO_CACHE — необязательный параметр. Не сохраняет результаты в кеше запросов.
SQL_CALC_FOUND_ROWS — необязательный параметр. Вычисляет, сколько записей находится в результирующем наборе (не принимая во внимание атрибут LIMIT), Количество записей можно получить с помощью функции FOUND_ROWS.
expressions — список возвращаемых столбцов (существующих и/или вычисляемых).
tables — таблицы, из которых требуетсяполучить записи (должна быть хотя бы одна таблицав предложении FROM).
WHERE conditions — необязательный параметр. Условия, которые должны быть выполнены для выбранных записей.
GROUP BY expressions — необязательный параметр. Данные собираются по нескольким записям, результаты группируются по одному или нескольким столбцам.

Выборка данных с помощью условий

Параметры SELECT

HAVING condition — необязательный параметр. Используется в сочетании с GROUP BY, для ограничения группы возвращаемых строк теми, где условие condition=TRUE.

ORDER BY expression — необязательный параметр. Используется для сортировки записей в результирующем наборе.

LIMIT — необязательный параметр. Используется для извлечения записей из одной или нескольких таблиц в MySQL и ограничения количества возвращаемых записей на основе предельного значения. Максимальное количество записей, заданных number_rows, будет возвращено в результирующем наборе. Первая строка, возвращаемая LIMIT, будет определяться значением offset_value.

PROCEDURE — необязательный параметр, proc_nm имя процедуры, обрабатывающей данные в результирующем наборе.

INTO — необязательный параметр. Позволяет записать результирующий набор в файл или переменную.

Выборка данных с помощью условий

Параметры SELECT

INTO — необязательный параметр. Опции INTO.
INTO OUTFILE ‘filename’ options. Записывает результирующий набор в файл с именем filename на хосте сервера. Для параметров можно указать:
FIELDS ESCAPED BY ‘character’
FIELDS TERMINATED BY ‘character’ [OPTIONALLY ENCLOSED BY ‘character’]
LINES TERMINATED BY ‘character’
где character — символ, отображаемый как символ ESCAPE, ENCLOSED или TERMINATED.
Пример.

```
SELECT supplier_id, supplier_name
FROM suppliers INTO OUTFILE ‘results.txt’
FIELDS TERMINATED BY ‘,’ OPTIONALLY ENCLOSED BY ‘»»‘
LINES TERMINATED BY ‘\n’;»
```

Выборка данных с помощью условий

Параметры SELECT

INTO — необязательный параметр. Опции INTO.
INTO DUMPFILE ‘filename’. Записывает одну строку набора результатов в файл с именем filename на хосте сервера. Не происходит прерывания столбца, не прерывается строка или обработка перехода.
INTO @variable1, @variable2,… @variable_n. Записывает набор результатов в одну или несколько переменных, указаннных в параметрах @ variable1, @ variable2, … @variable_n

Работа с транзакциями.
FOR UPDATE — необязательный параметр. Записи, затронутые запросом, блокируются, пока транзакция не завершится.
LOCK IN SHARE MODE — необязательный параметр. Записи, затронутые запросом, могут использоваться другими транзакциями, но не могут быть обновлены или удалены этими и другими транзакциями.

Операторы сравнения.

Операторы сравнения = и !=

Получить фамилии всех студентов, которые не

Алексеев А. А

```
SELECT Name FROM Student WHERE Name != ' Алексеев А. А ';
```

Операторы больше и меньше

Получить фамилии всех студентов, номер личного дела которых больше 1233.

```
SELECT Name FROM Student WHERE StudentID > 1233;
```

Операторы <= и >=

Получить фамилии всех студентов, номер личного дела которых не больше 1233 и не меньше 1234.

```
SELECT Name FROM Student
                    WHERE StudentID<=1233 AND StudentID>=1234;
```

Поиск текстовых данных по шаблону

В рассмотрим *поиск текстовых данных по шаблону* с помощью предложения where и оператора LIKE.

Оператор сравнения на равенство (=) выбрает одинаковые строки

SELECT *Name* from *Student* where *Name* = 'Алексеев А. А ';

Надо вывести данные о студентах, имя которых начинается с буквы А Язык SQL позволяет выполнить поиск строковых данных по шаблону.

Для этого в предложении where используется оператор LIKE следующим образом.

SELECT *Name* from *Student* where *Name* LIKE ' A%';

Знак % действует как символ-заместитель (аналогично использованию * в системах DOS и Linux).

"B%" обозначает все строки, которые начинаются с буквы B.

"%A" обозначает строки, которые заканчиваются символом A

"%A%" обозначает строки, которые содержат букву A.

Внешние ключи FOREIGN KEY

Foreign Keys является способом обеспечить целостность данных БД.
Внешний ключ означает, что значения в одной таблице должны также появиться в другой таблице.
Ссылающаяся таблица называется parent table (родительской таблицей).
Таблица с **foreign key** (внешним ключом) называется child table (дочерней таблицей).
Foreign key в дочерней таблице, как правило, ссылаются на primary key (первичный ключ) в родительской таблице (или уникальный столбец).
Foreign key может быть определен либо в операторе CREATE TABLE или в операторе ALTER TABLE.

Синтаксис.

```
CREATE TABLE tbl_nm
(col1 datatype null/not null, col2 datatype null/not null,…
CONSTRAINT fk_col FOREIGN KEY (col1, coln2, … col_n)
REFERENCES parent_tble (col1, col2, … coln_n));
#fk_col – уникальное имя ограничения,
#  пишется в кавычках ``(key ~)
```

Пример 3.

Ограничения внешних ключей

```
CREATE TABLE  Subject1
(  SubID INT NOT NULL AUTO_INCREMENT PRIMARY KEY,
  Name VARCHAR(80),  TimeHr INT
) ENGINE= InnoDB;

CREATE TABLE  Student_Subj
(
 St_SubID INT NOT NULL AUTO_INCREMENT PRIMARY KEY,
 StudentID INT NOT NULL, SubID INT,Est INT,
 CONSTRAINT `Sub` FOREIGN KEY(SubID)  REFERENCES
                                          Subject1(SubID)
                    ON DELETE CASCADE ON UPDATE CASCADE,
 CONSTRAINT `St` FOREIGN KEY(StudentID) REFERENCES
                                          Student(StudentID)
                    ON DELETE CASCADE ON UPDATE CASCADE
) ENGINE= InnoDB;
```

```
INSERT INTO Subject1(SubID , Name,  TimeHr) VALUE
 (11,'Math.analis1',  51);
INSERT INTO Subject1(Name,  TimeHr) VALUE
 ('Math.analis',  310),('Lin.Algebra',170), ('Discret.math',85),
('TFCV',85); #значения  SubID 11,12,13,14,15

SELECT * FROM Subject1; #вывести все (*) значения из таблицы

INSERT INTO Student_Subj(StudentID,SubID , Est) VALUE
 (1232,11,5);
INSERT INTO Student_Subj(StudentID,SubID , Est) VALUE
(1233,15,3);
INSERT INTO Student_Subj(StudentID,SubID , Est) VALUE
(1233,13,3);
SELECT * FROM Student_Subj;
```

```
INSERT INTO Student_Subj(StudentID,SubID , Est)
            VALUE (1231,17,3);
```

Ошибка

```
  (1231,17,3)Cannot add or update a child row:
 # a foreign key constraint fails (`students1/student_subj`,
 #CONSTRAINT `St` FOREIGN KEY (`StudentID`) REFERENCES `student`
(`StudentID`)  # ON DELETE CASCADE ON UPDATE CASCADE)

  UPDATE Subject1  SET SubID=17  WHERE SubID=13;
  SELECT * FROM Student_Subj;

  UPDATE Student_Subj  SET SubID=13  WHERE
  St_SubID=3;
 ALTER TABLE Student_Subj ADD  UNIQUE
 (StudentID,SubID);
 describe Student_Subj;
INSERT INTO Student_Subj(StudentID,SubID , Est)  VALUE
(1233,15,3);
```

Duplicate entry '1233-15' for key 2

Задание 2.

Повторить **Пример 2** (слайд № 68) и **Пример 3** (слайды № 74-76)**.**