Поиск

4

Не заполнены данные профиля педагога. Заполнить данные профиля.

Вы не установили свою фотографию. Установить её вы можете на своей странице. Установить фотографию.

Хотите получать уведомления от нашего сайта?

Включить Не сейчас



# Базы данных

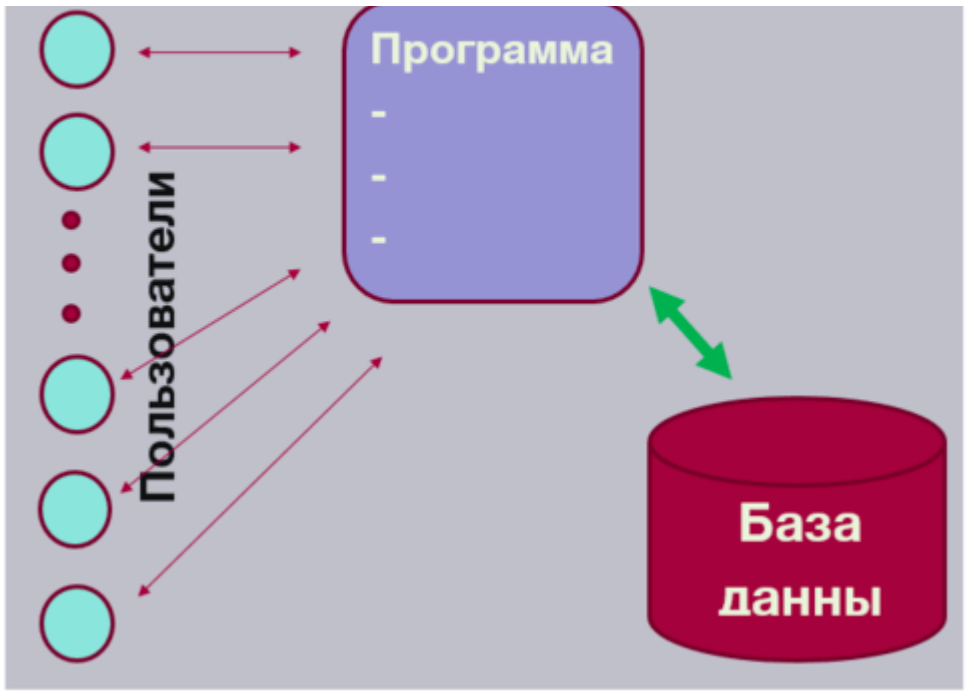
Образовательный уровень: 1

## Оглавление



## Урок 1. Введение.

Мы начинаем новый курс – Базы дынных. Специалисты, работающие с базами данных одни из самых востребованных и высокооплачиваемых в наше время. Вы наверняка знаете такие приложения, как Instagram, Вконтакте, Like, TikTok, Youtube, игровые программы: ConterStrike. BrawlStars, WorldofTanks  - все они используют базы данных. Схематически это выглядит так (рисунок 1):




По сути, база данных – это файлы на жестком диске (для тех, кто не знает можно вкратце объяснить, что такое жесткий диск, какую роль он играет).
Эти файлы хранят различные данные для программы. Например Youtube – это программа, он работает с базой данных, в которой хранятся видеоролики, данные аккаунтов, данные каналов, комментарии. Также в базе данных хранятся не только сами данные, но и информация о их взаимосвязях – какой ролик к какому аккаунту привязан, какой комментарий к какому ролику относиться и от какого пользователя оставлен.
Мы с вами научимся с помощью языка Python создавать базу данных, добавлять, удалять, редактировать, получать данные. Создавать структуру данных. Также изучим специальный язык запросов SQL, с помощью которого можно производить различные действия с данными.

Что собой представляет база данных? Практически все современные базы данных представляют собой набор таблиц, связанных между собой.
Рассмотрим на примере:
Представим, что нам нужно создать базу данных фильмов.
Значит в нашей базе данных (БД) должны быть таблица с информацией о фильмах.
@Какие данные вы бы хранили о фильмах

! Проблемная задача
Выбрать на свое усмотрение объекты, информацию о которых можно хранить в базе данных, составить таблицу с данными об этих объектах.

**ФИЛЬМЫ**

| Название | Жанр | Год выхода | Актер | Режиссер |
|---|---|---|---|---|
| Человек-паук | Фантастика | 2002 | Тоби Магуайр | Сэм Рэйми |
| Побег из Шоушенка | Драма | 1994 | Тим Роббинс | Фрэнк Дарабонт |
| 1 + 1 | Комедия | 2011 | Франсуа Клюзе | Оливье Накаш |

Перейдем к практике.
Для работы с базой данных в Python используется библиотека sqlite3.
Импортируем ее:
```
import sqlite3
```

Чтобы создать базу данных используется команда **connect(**путь**):**
```
con = sqlite3.connect("filmdatabase.db")
```

В скобках указывается путь к файлу базы данных. Как обычно, если файл находится в той же директории, где и программа, то достаточно указать имя файла, иначе необходимо прописать полный путь.

Команда **connect** по сути является функцией и возвращает объект класса **Connection**, мы его назвали – **con** (сокращение от слова **connection**). Этот класс описывает как бы не саму базу данных, а соединение с ней. И с помощью этого соединения – объекта класса **Connection** мы, как бы получаем доступ к базе данных и возможности ее редактировать и получать данные.
Если мы сейчас запустим программу:
```
import sqlite3

con = sqlite3.connect("filmdatabase.db")
```

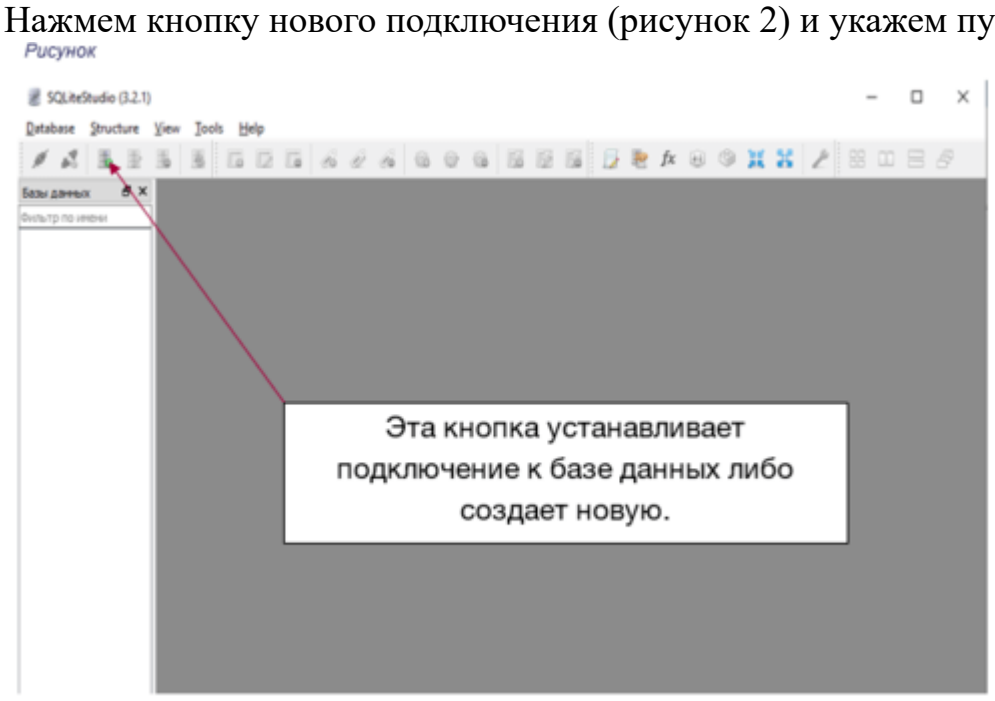
**mmdatabase.db**

Все сработало.
! Проблемная задача
Создать две базы данных: myfriends.db и games.db

Мы научились с помощью Python создавать базу данных. Пока эта база данных пустая. В ней нет таблиц.
Дальше мы будем создавать таблицы, вносить в них данные, получать и редактировать их. Так как мы будем делать все это программируя на python, то было бы удобно с помощью какой-нибудь специальной программы просматривать нашу базу данных. Такая программа называется:
**SQLiteStudio**
Как видите у нее часть названия совпадает с названием модуля sqlite3.
Установим эту программу с официального сайта: https://sqlitestudio.pl
На главной странице сайта жмем кнопку Download и программа загружается.
Загруженный архив распаковываем и заходим в распакованную директорию.
Устанавливать ничего не надо.
Далее запускаем файл программы **SQLiteStudio.exe**

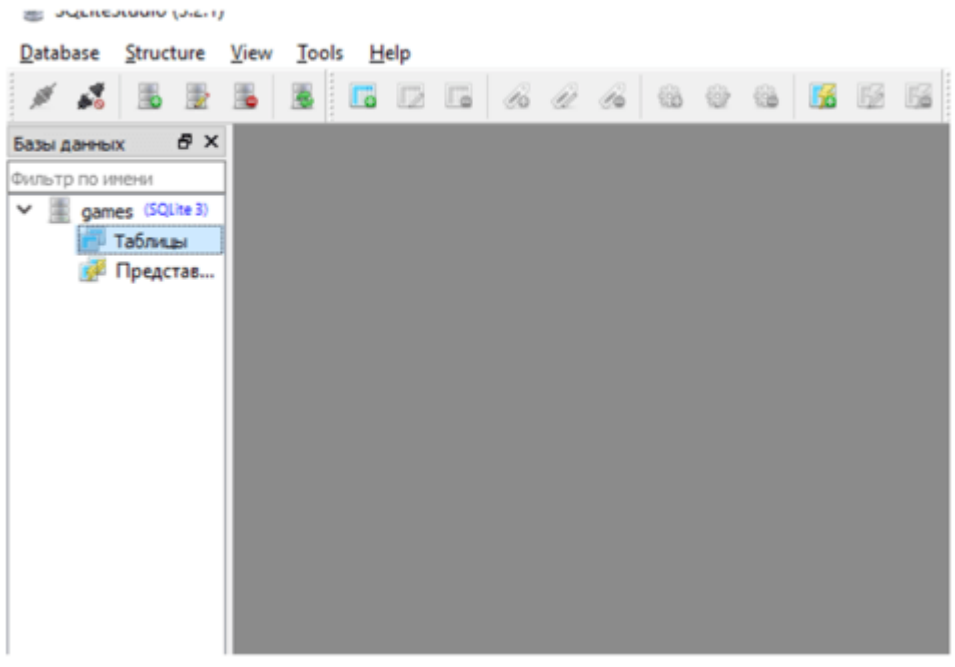
Нажмем кнопку нового подключения (рисунок 2) и укажем путь к нашей базе данных, которую мы создали.

*Рисунок*




В результате в левой панели появится имя нашей базы данных (рисунок 3):

*Рисунок*

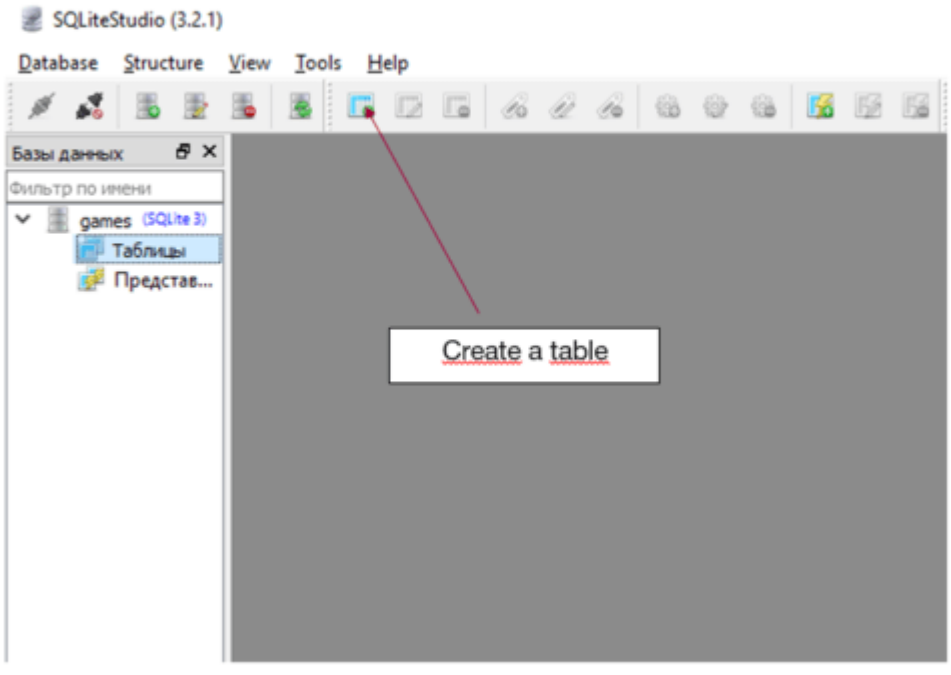



Теперь мы можем с помощью программы **SQLiteStudio** просматривать нашу базу данных, и не только. Также производить с ней любые действия.
Это удобно для просмотра результатов работы наших программ.

! Проблемная задача
Попробовать самостоятельно создать таблицу в программе **SQLiteStudio** для своей базы данных.
Добавить в нее 10-15 записей.

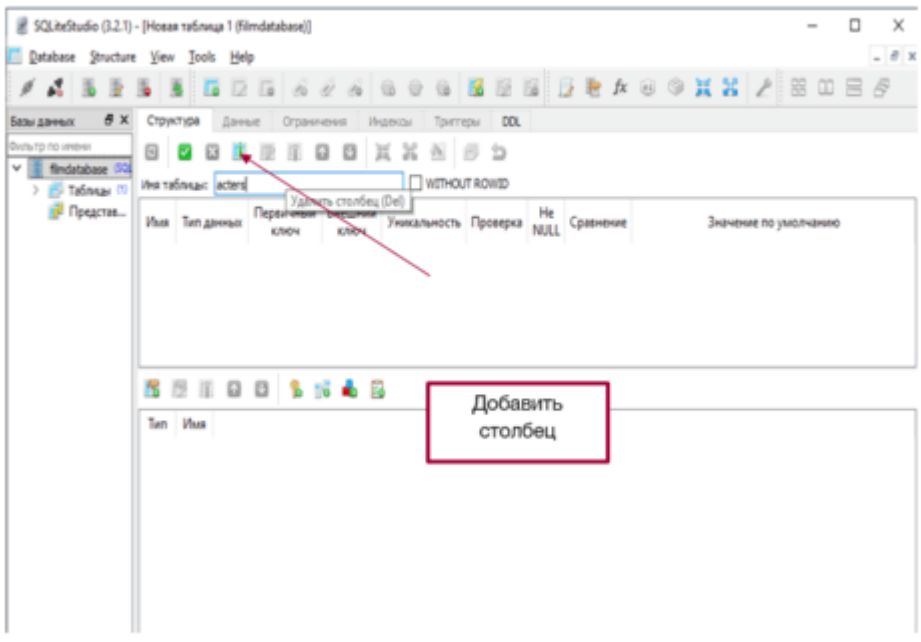
Решение:

Нажимаем кнопку **Create a table** (рисунок 4)




Далее вводим имя таблицы и создаем столбцы. Чтобы добавить столбец в таблицу необходимо нажать кнопку **Добавить столбец** (рисунок 5)

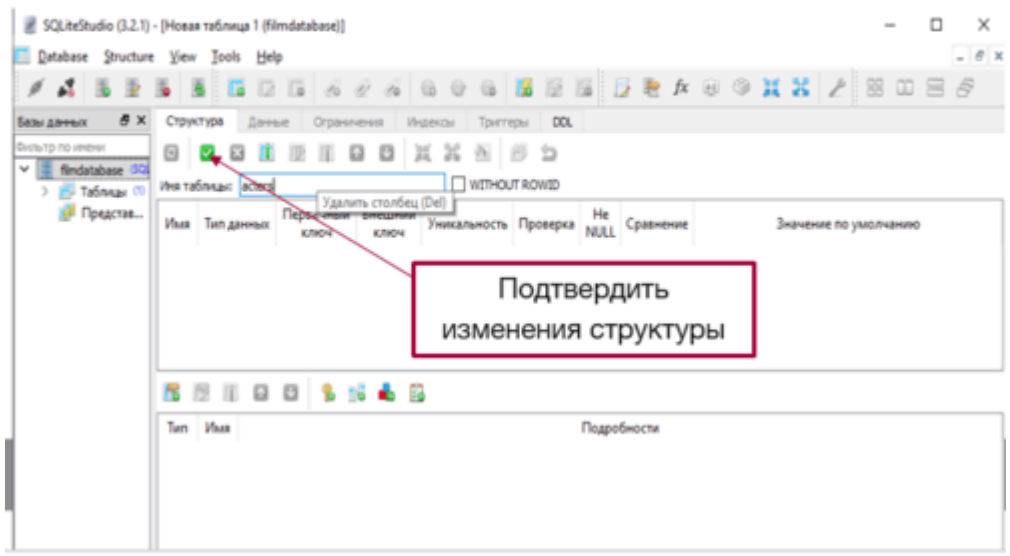
*Рисунок*




В открывшемся окне вводим название столбца и выбираем тип данных, которые будут отображаться в этом столбце. Остальные настройки будем проходить позже, их пока не трогаем.

После того, как создали все столбцы, нажимаем кнопку для сохранения изменений в базе данных (рисунок 6):

*Рисунок*




После нажатия этой кнопки, откроется окно: **Запросы, которые будут выполнены**
В этом окне будет отображаться текст SQL запроса:

```
CREATE TABLE acters (
Имя TEXT,
Возраст INTEGER
);
```

любой БД.

# Урок 2. Создание таблиц.

На прошлом уроке мы познакомились с программой **SQLiteStudio**, открыли с помощью нее нашу БД, которую мы создали на Python.
Далее вручную с помощью **SQLiteStudio** создали в нашей БД таблицу и столбики в ней.
И в момент сохранения изменений у нас отобразилось окно с SQL запросом:

```
CREATE TABLE acters (
Имя TEXT,
Возраст INTEGER
);
```

Начало изучения языка запросов SQL начнем с разбора данного запроса.
Этот запрос создает новую таблицу. Как видно для этого используется команда:
CREATE TABLE имя
имя - это имя создаваемой таблицы.
Далее в скобках перечисляются через запятую названия столбцов, рядом с каждым названием указывается тип данных этого столбца.
В нашем случае создается таблица с названием acters. В ней будет два столбца: 1. Столбец с именем **Имя** и типом данных **TEXT** (текстовый).

Нам нужно понять, как такой запрос выполнить в Python.
На прошлом уроке мы создали объект – подключение к базе данных. С помощью метода **cursor()** этого объекта необходимо создать объект-курсор, который как раз и будет выполнять SQL запросы.
Создаем курсор:

```
import sqlite3

con = sqlite3.connect("filmdatabase.db")
cursor = con.cursor()
```

И теперь, чтобы выполнить SQL запрос, необходимо вызвать метод **execute(sql)** у объекта cursor, а скобках передать параметр **sql**, который по сути является обычной строкой с SQL запросом.

! Проблемная задача:
Создать в базе данных таблицу с тремя столбцами с помощью Python.

Теперь узнаем, как нам добавлять данные в таблицу с помощью SQL запроса.
Делается это с помощью команды:
INSERT INTO имя_таблицы VALUES (val1, val2, …)
Где в скобках перечисляются значения, в таком же порядке как идут соответствующие столбцы.
Разберем на примере:
Чтобы добавить в таблицу **acters**:

| **Имя** | **Возраст** |
|---|---|
| | |

Новую запись:
**Эдди Мерфи 59 лет**
Нужно выполнить SQL запрос:
INSERT INTO acters VALUES (‘Эдди Мерфи’, 59)

```
cursor.execute('INSERT INTO acters VALUES ("Эдди Мерфи", 59)')
con.commit()
```

Метод **commit**() вносит изменения в базу данных. Когда выполнилась команда добавления записи в нашу таблицу с актерами, эти изменения
произошли пока только в оперативной памяти. Чтобы эти изменения произошли непосредственно в файле базы данных на жестком диске, необходимо выполнить метод **commit**() у объекта **con**.

! Проблемная задача:
Добавить в программу код, добавляющий в таблицу в нашей БД 5 новых записей.

! Проблемная задача

На python:
Создать новую таблицу с 5 полями, добавить в нее 5 записей, данные программа должна запросить у пользователя.

# Урок 3. SELECT.

На этом уроке разберем, как можно получать данные из БД с помощью SQL запроса.

На прошлом уроке мы уже немного заполнили нашу БД данными. Теперь получим их в **python** с помощью SQL запроса.
Для этого используется оператор SELECT:
SELECT col1, col2, … FROM table
Данная команда получает данные из таблицы, которые содержаться в столбцах, перечисленных после оператора SELECT col1, col2, …

Проверим на практике и выполним SQL запрос с оператором SELECT.
cursor.execute('SELECT имя, возраст FROM acters')

Но как нам эти данные увидеть, сохранить куда-нибудь?
Для этого у объекта cursor есть два метода:
**fetchone**() – возвращает одну запись (строку из таблицы) в виде кортежа, если вызвать этот метод повторно, то он вернет следующую запись.
**fetchall**() – получает все записи (строки таблицы) в виде списка, элементами которого являются кортежи.

**Примечание**:
Кортеж – это список, который нельзя изменять. Вместо квадратных скобок используются круглые.
Пример кортежа с данными человека (Иван, возраст 16 лет, из России):
T = (‘Ivan’, 16, ‘Russia’)

! Проблемная задача.
Протестировать работу методов **fetchone**() и **fetchall**() – вывести на экран кортежи и список кортежей, которые возвращают эти методы.

Решение:

```
cursor.execute('SELECT имя, возраст FROM acters')
print(cursor.fetchone())
print(cursor.fetchone())

cursor.execute('SELECT имя, возраст FROM acters')
print(cursor.fetchall())

cursor.execute('SELECT имя, возраст FROM acters')
for row in cursor.fetchall():
    print(row[0], row[1])
```

Как видно, мы несколько раз выполняли SQL запрос SELECT. Это потому, что после выполнения метода **fetchone**() или **fetchall**(), полученная запись или записи как бы удаляются из курсора (cursor) и их необходимо заново получать, если нужно повторно использовать. Либо сохранять в кортеж или список при первом использовании.

! Проблемная задача:
Добавить в таблицу актеров несколько записей, чтобы их количество было не меньше 10.
Составить программу, которая получает все записи о актерах из базы данных и выводит их на экран в отсортированном по возрасту формате. По возрастанию.

# Урок 4. ИНТЕРФЕЙС.

На этом уроке мы будем использовать знания, полученные на предыдущем курсе, а именно графический интерфейс. Намного лучше работать с данными, которые отображаются в удобном виде.

Импортируем сразу все из библиотеки **tkinter**:
from tkinter import *

Создадим окно приложения:
```
window = Tk()
window.geometry('300x300+100+100')
window.title("work with database")
window.mainloop()
```

```
def getrow():
    cursor.execute('SELECT имя, возраст FROM acters')
    row = str(cursor.fetchone())
    Linfo.configure(text = row)

#connect BD
con = sqlite3.connect("filmdatabase.db")
cursor = con.cursor()

#interface
window = Tk()
window.geometry('300x300+100+100')
window.title("work with database")
Linfo = Label(window, text = 'Запись')
Linfo.pack()

B = Button(text="Получить запись", width=15, height=2, command = getrow)
B.pack()

window.mainloop()
```

Как видно, мы применили функцию **str**() к результату метода **fetchone**(), и у нас при нажатии кнопки текст Label стал таким:
(‘Эдди Мерфи’, 59)
Python перевел кортеж в строку в явном виде, с круглыми скобками и поля имеющие строковый тип отобразил с кавычками.

! Проблемная задача:
Отобразить полученную запись в читаемом формате, без скобок и кавычек.

Решение:

```
def getrow():
    cursor.execute('SELECT имя, возраст FROM acters')
    row = cursor.fetchone()
    text = ''
    for el in row:
        text = text + str(el) + ' '
    Linfo.configure(text = text)
```

Возможны и другие варианты решения.

Мы научились получать запись из таблицы базы данных и отображать ее в нашем окне приложения.
Но, хотелось бы отобразить не одну запись, а все (все строки из таблицы).

В этом нам поможет виджет **Listbox**. В нем можно отображать список строк. С помощью метода **insert**() в него добавляется новая строка.
Рассмотрим на примере:

```
def getrow():
    cursor.execute('SELECT имя, возраст FROM acters')
    rows = cursor.fetchall()
    for row in rows:
        text = ''
        for el in row:
            text = text + str(el) + ' '
        LBinfo.insert(END, text)
…
…
…


LBinfo = Listbox(window, width = 30)
LBinfo.pack()
```

В методе **insert**() используется слово-константа **END –** оно обозначает, что вставляться строка будет в конец списка в **Listbox**.

! Проблемная задача:

Проверить работу с помощью получения данных из БД и отображения их в виджете Listbox.

# Урок 5. ЗАПРОСЫ с УСЛОВИЕМ. Удаление и обновление записей.

На прошлом уроке мы написали программу, с помощью которой можно получать, отображать и вносить данные в базу данных.

Но, когда мы получали данные с помощью SQL запроса SELECT, то мы получали все строки из таблицы. А если нам нужно получить не все строки, а, например, нам нужно получить список актеров, возраст которых меньше 35 лет.

Для этого в операторе SELECT предусмотрен оператор WHERE, с помощью которого можно задать условие для какого-либо параметра.

Например, для получения записей из таблицы актеров с возрастом меньше 35 лет SQL запрос будет выглядеть вот так:

SELECT имя, возраст FROM acters WHERE возраст<35

Проверим работу запроса в программе.

! Проблемная задача:
Добавить столбец в таблицу актеров – фильм, в котором играл актер.

! Проблемная задача:
Добавить в интерфейс программы возможность выбора условия для выводимых записей:
- условие на возраст
- условие на имя
- условие на фильм (чтобы можно было вывести список актеров сыгравших в каком-то определенном фильме
Чтобы удалить запись из таблицы используется оператор DELETE.

Пример:

DELETE FROM <имя таблицы >
WHERE <условие>

Мы удаляем те записи (строки), которые соответствуют определенному условию.

Чтобы обновить данные в одной или нескольких записях, используется оператор UPDATE.

Пример:

**UPDATE** acters

**SET** film = ‘Терминатор, age = 38

**WHERE** name = ‘Arnold’

Данный запрос найдет запись в таблице актеров с именем **Arnold** и изменит значения в столбцах **film** и **age** на значения **Терминатор** и **38** соответственно.

! Проблемная задача:

Добавить в программу возможность удалять запись по заданному значению поля.

! Проблемная задача:
Добавить в программу возможность обновлять данные об указанном пользователем актере. Также пользователь указывает новые значения полей.

# Урок 6. Нормальные формы 1 и 2

На прошлых уроках мы научились на языке Python создавать базу данных, создавать в ней таблицы, добавлять записи в таблицу, удалять и редактировать. Также научились запрашивать данные из базы данных. Но, оказывается, для полноценной работы с базой данных этих навыков недостаточно. Чтобы все правильно работало, база данных, а точнее, ее структура должна соответствовать нескольким условиям.
Рассмотрим их.

- Ученики, их оценки по предметам в четвертях, кто в каком классе учиться.
- Информация о учителях и, соответственно о том, у какого учителя какие классы и какие предметы.

! Проблемная задача:
Составить таблицу, которая будет являться базой данных для школы и содержать все перечисленные данные. Добавить в нее 5-7 записей.

Решение:

Смысл в том, что в одной таблице невозможно отобразить такие данные, это будет выглядеть очень громоздко и непонятно. Что-то наподобие этого:

**База Данных Школа**

| Ученик | Класс | Предметы | Учителя | Оценки |
|---|---|---|---|---|
| Иванов Иван | 4а | Математика, Русский язык, Окружающий мир, Физкультура | Природин Т.Р., Приставкина Е.У., Бегунов Д. С., Циркулев А.В. | Математика, 5, 1 четверть, Математика, 4, 2 четверть….Русский язык, 4, 1четверть…. |
| Петров Андрей | 10Б | Алгебра, Русский язык, Геометрия, Физкультура, География, Английский язык | Сажин Т.Л., Понарин А.В., Пичугин П.Р., Назаров Л.Б. | Геометрия, 5, 1 четверть, Алгебра, 4, 2 четверть…. Русский язык, 4, 1четверть…. |

…

# Или так:

**БД Школа**

| Ученик | Класс | Предметы | Учителя | Оценки | Четверть | Классный руководитель |
|---|---|---|---|---|---|---|
| Иванов Иван | 4а | Русский язык | Приставкина Е.У. | 4 | 1 | Симакова Д.Н. |
| Иванов Иван | 4а | Русский язык | Приставкина Е.У. | 3 | 2 | Симакова Д.Н. |
| Иванов Иван | 4а | Русский язык | Приставкина Е.У. | 5 | 3 | Симакова Д.Н. |
| … | … | … | … | … | … | … |
| Иванов Иван | 4а | Математика | Циркулев А.В. | 5 | 1 | Симакова Д.Н. |
| Иванов Иван | 4а | Математика | Циркулев А.В. | 4 | 2 | Симакова Д.Н. |
| … | … | … | … | … | … | … |

*У учеников могут получиться разные результаты, и возможно кто-то сразу сделает несколько таблиц, а кто-то в одной таблице все представит большим количество записей, чтобы в каждой ячейке было одно значение. Это будут верные решения.*

Как видно в таблице есть ячейки, содержащие не одно какое-то значение, а множество значений. Например, в столбце **Предметы** по конкретному ученику в одной ячейке содержаться несколько названий предметов. И если нам нужно составить отчет по ученикам, которые учат **Английский язык**, то нужно будет загрузить все записи из таблицы, с помощью метода **fetchall**() сохранить их в список, потом осуществить перебор элементов списка, и в каждом списке проверить элемент содержащий данные о предметах (а этот элемент по своей структуре тоже

переписывать практически заново.
Поэтому можно обозначит такой принцип структуры базы данных:
На пересечении столбца и строки в любой таблице должно быть одно значение (не состоящее из нескольких элементов, которые тоже могут иметь разные значения, как например список).

Помимо этого принципа, который мы свами сами вывели, есть еще такой - в таблице не должно быть повторяющихся строк (это логично, иначе невозможно будет точно определить, что это именно та запись, которая нужна)

Например, в нашей таблице если в каком-то классе будут полные тезки, то будет много повторяющихся записей. Эту проблему решает такое условие:
В каждой таблице должен быть первичный ключ.
Первичный ключ – это набор полей (столбцов или атрибутов), который не повторяется ни у одной записи.
Если добавить поле – дата рождения, то и это не решит проблему до конца (вдруг будет два ученика с одинаковыми ФИО и датой рождения, это конечно маловероятно, но возможно, например из-за ошибки ввода), поэтому принято вводить искусственное поле (столбец) – **id**, значением которого является уникальный номер записи, который не повторяется ни у одной записи (строки) таблицы.
Если база данных соответствует этим условиям (принципам), то говорят, что она приведена к **1 (первой) нормальной форме:**

**БД Школа (первая нормальная форма)**

| id | Ученик | Класс | Предметы | Учителя | Оценки | Четверть | Классный руководитель | |
|---|---|---|---|---|---|---|---|---|
| 1 | Иванов Иван | 4а | Русский язык | Приставкина Е.У. | 4 | 1 | Симакова Д.Н. | |
| 2 | Иванов Иван | 4а | Русский язык | Приставкина Е.У. | 3 | 2 | Симакова Д.Н. | |
| 3 | Иванов Иван | 4а | Русский язык | Приставкина Е.У. | 5 | 3 | Симакова Д.Н. | |
| … | … | … | … | … | … | … | … | |
| 234 | Иванов Иван | 4а | Математика | Циркулев А.В. | 5 | 1 | Симакова Д.Н. | |
| 235 | Иванов Иван | 4а | Математика | Циркулев А.В. | 4 | 2 | Симакова Д.Н. | |
| … | … | … | … | … | … | … | … | |

Есть и **2 (вторая) нормальная форма.**
Если взять нашу таблицу с учениками, где уже в каждой ячейке расположено по одному значению, то видно, что в таблице очень много повторяющихся данных. Такая база данных будет занимать очень много памяти. Это очень неэффективно, да и тоже сложновато как-то для поиска. Поэтому придумали такое правило:
Данные, которые не зависят от первичного ключа должны быть вынесены в отдельную таблицу.

Например, в нашей таблице от первичного ключа не зависит значение в поле **Предметы**. Значение этого поля не зависит от того, в какой оно строке таблицы, так как у одного ученика несколько предметов. Поэтому вынесем его в отдельную таблицу.

! Проблемная задача:
Вынести в отдельную таблицу данные о **Предмете.**

Решение:

**БД Школа**

| id | Ученик | Класс | Предмет | Учитель | Оценки | Четверть | Классный руководитель | |
|---|---|---|---|---|---|---|---|---|
| 1 | Иванов Иван | 4а | 1 | Приставкина Е.У. | 4 | 1 | Симакова Д.Н. | |
| 2 | Иванов Иван | 4а | 1 | Приставкина Е.У. | 3 | 2 | Симакова Д.Н. | |
| 3 | Иванов Иван | 4а | 1 | Приставкина Е.У. | 5 | 3 | Симакова Д.Н. | 10:35 |
| … | … | … | … | … | … | … | … | … |
| 234 | Иванов Иван | 4а | 2 | Циркулев А.В. | 5 | 1 | Симакова Д.Н. | 11:30 |
| 235 | Иванов Иван | 4а | 2 | Циркулев А.В. | 4 | 2 | Симакова Д.Н. | 10:35 |
| … | … | … | … | … | … | … | … | … |

| id | Предмет |
|---|---|
| 1 | Русский язык |
| 2 | Математика |
| 3 | Алгебра |
| 4 | Геометрия |
| 5 | Окружающий мир |
| 6 | География |
| 7 | Информатика |
| | … |

! Проблемная задача:
Определить какие еще поля в БД Школа не соответствуют 2 (второй) нормальной форме.
Привести БД Школа ко 2 (второй) нормальной форме.

Решение:

| id | Ученик | Класс | Учителя | Предмет | Оценки | Четверть | Классный руководитель | |
|---|---|---|---|---|---|---|---|---|
| **1** | Иванов Иван | 4а | 1 | 1 | 4 | 1 | Симакова Д.Н. | |
| **2** | Иванов Иван | 4а | 1 | 1 | 3 | 2 | Симакова Д.Н. | |
| **3** | Иванов Иван | 4а | 1 | 1 | 5 | 3 | Симакова Д.Н. | |
| … | … | … | … | | … | … | … | |
| **234** | Иванов Иван | 4а | 2 | 2 | 5 | 1 | Симакова Д.Н. | |
| **235** | Иванов Иван | 4а | 2 | 2 | 4 | 2 | Симакова Д.Н. | |
| … | … | … | … | … | … | … | … | |

| id | Предмет |
|---|---|
| 1 | Русский язык |
| 2 | Математика |
| 3 | География |
| 4 | Алгебра |
| | … |

| id | Учитель |
|---|---|
| 1 | Приставкина Е.У. |
| 2 | Циркулев А.В. |
| 3 | Природин Т.Р |
| 4 | Сажин Т.Л |
| | … |

Какие можно отметить плюсы такой формы?
- если изменить название какого предмета, то в таблице учеников ничего менять не придется, так как в поле **Учитель** указан номер строки в таблице учитель, а номер, несмотря на то что изменилось название предмета, остался старым.
- База данных стала занимать меньше памяти, раньше в каждой строке таблицы с учениками в поле **Предмет** хранилось название предмета, а это конечно больше занимает памяти чем целое число – **id предмета**.

# Урок 7. Нормальная форма 3

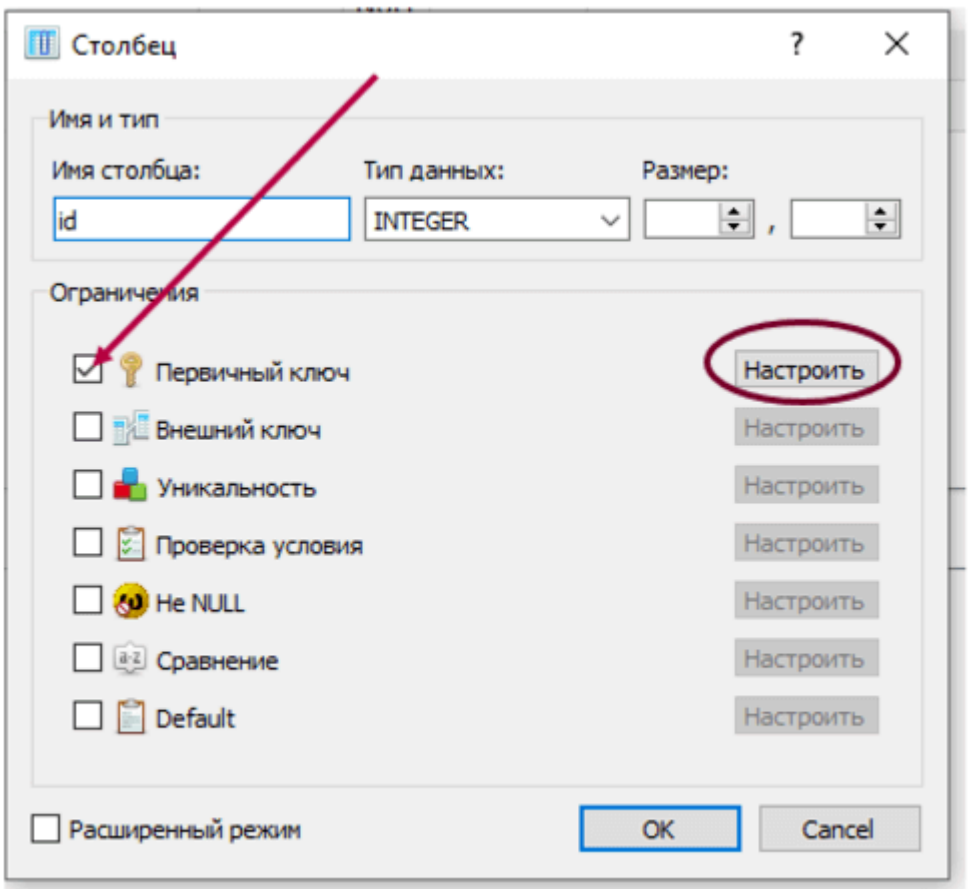
! Проблемная задача:
Привести ко 2 (второй) нормальной форме ранее созданную БД, либо создать новую. Внести изменения в программу на Python, которая позволяет управлять данными в БД (добавлять, удалять, редактировать и просматривать).

Она звучит так:
БД соответствует **третьей нормальной форме**, если, во-первых, она соответствует второй нормальной форме, и плюс к этому в таблицах базы данных отсутствует **транзитивная зависимость**.
Транзитивная зависимость – это когда одно поле определяет второе, а второе определяет третье, но второе не определяет первое. В общем, звучит все страшно, но ничего сложного. Рассмотрим на примере нашей БД Школа.
Поле **Ученик** однозначно определяет поле **Класс.** На самом деле, если взять любую строку из таблицы Ученики, то имя ученика точно определяет его класс, одни ученик не может быть сразу в двух классах, такого не бывает. По ученику мы сразу можем сказать какой у него класс.
А вот поле Класс в свою очередь однозначно определяет поле Классный руководитель. Действительно, у одного класса не может быть двух классных руководителей.
Получается, что у нас в таблице учеников транзитивная зависимость:
**Ученик -> Класс -> Классный руководитель**
Чтобы устранить такую зависимость, можно создать отдельную таблицу классных руководителей и таблицу классов, а также убрать из таблицы учеников поле Классный руководитель.
! Проблемная задача:
Привести БД Школа к третьей нормальной форме (выполнить действия, описанные в предыдущем абзаце).

! Проблемная задача: внести изменения в программу.

Еще раз поговорим о первичном ключе. На прошлом уроке мы определили, что первичный ключ - это набор полей (столбиков) значения в которых не повторяются в таблице. Также мы сказали, что принято в качестве первичного ключа создавать искусственное поле **id.** Далее мы увидим, что не во всех таблицах нужно создавать такое поле.

А пока, в SQLiteStudio сделаем это поле первичным ключом, для этого установим галочку:




Затем нажмем кнопку **настроить** и выберем опцию **Автоинкремент**.

Эта опция делает так, что поле **id** будет заполняться автоматически.

! Проблемная задача:
Задач первичные ключи для остальных таблиц в БД Школа.

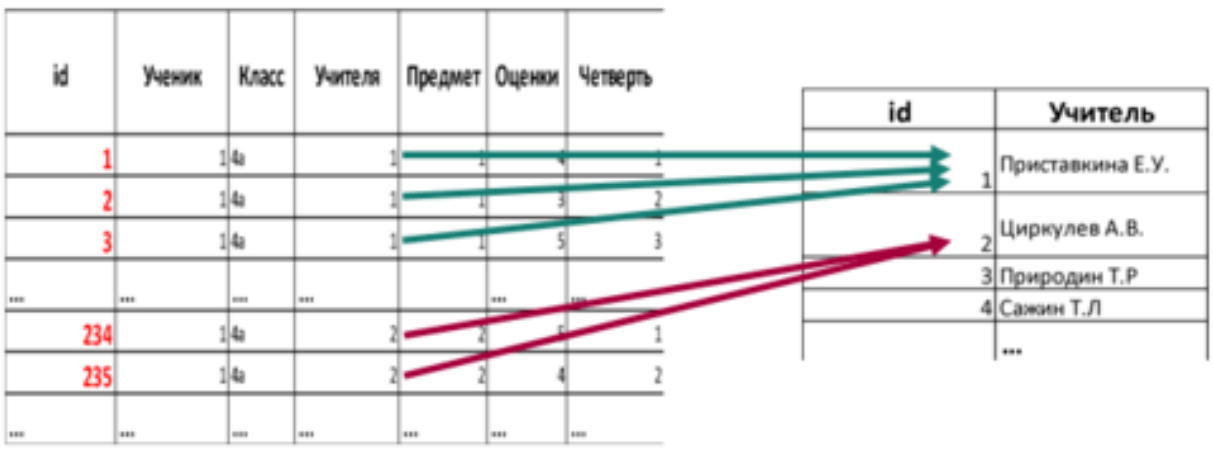
# Урок 8. Внешний ключ

Итак, на данный момент мы имеем базу данных, которая приведена к 3-ей нормальной форме. Можно сказать, что у нас довольно оптимальная база данных, все разделено по табличкам, лишней памяти поля не занимают и структура достаточно логичная. Хотя, конечно, можно и еще дальше улучшать.
Но, когда мы выполняем запрос все записей из таблицы **Ученики**, мы получаем в полях **Учитель** и **Предмет** числа вместо имен и названий.

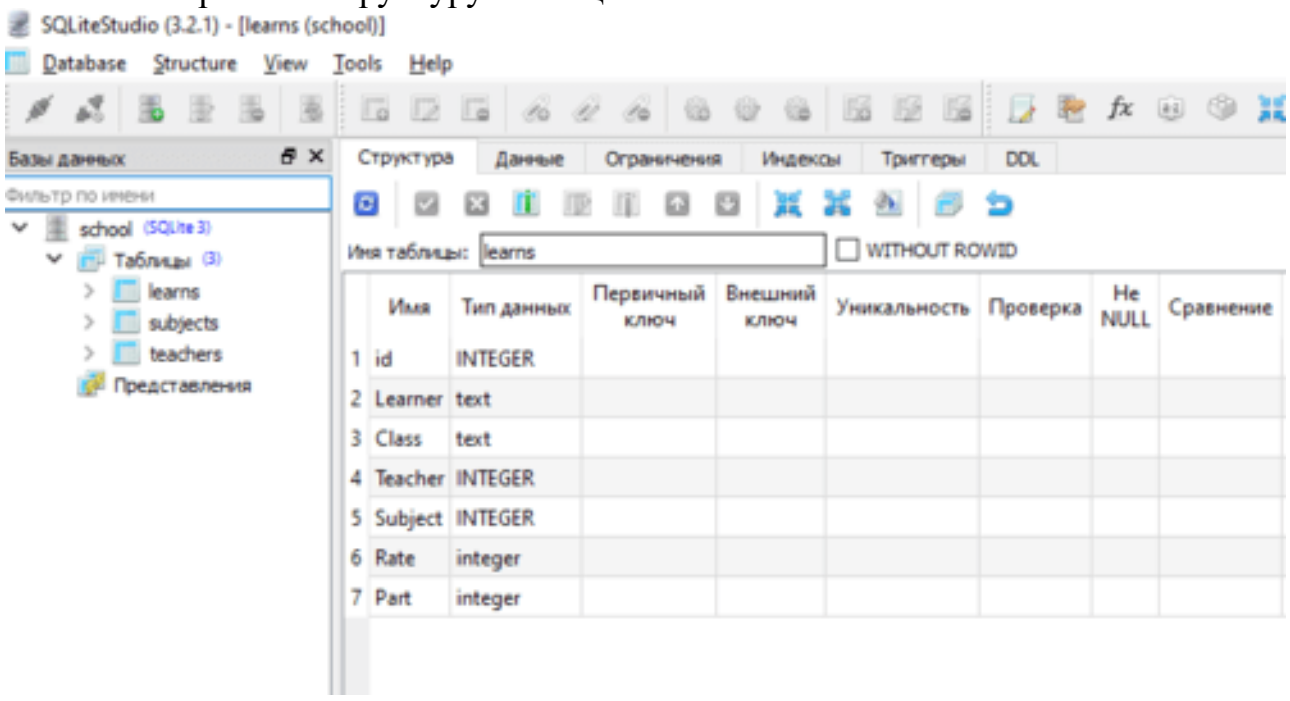
Можно, конечно, вручную загрузить все записи из таблицы **Учитель** в список, затем в списке с записями из таблицы **Ученики** заменить числа в поле **Учитель** на соответствующее этому номеру имя из списка с учителями.

! Проблемная задача:
Создать функцию, которая заменяет в списке с учителями числа в полях **Учитель** на имена учителей.

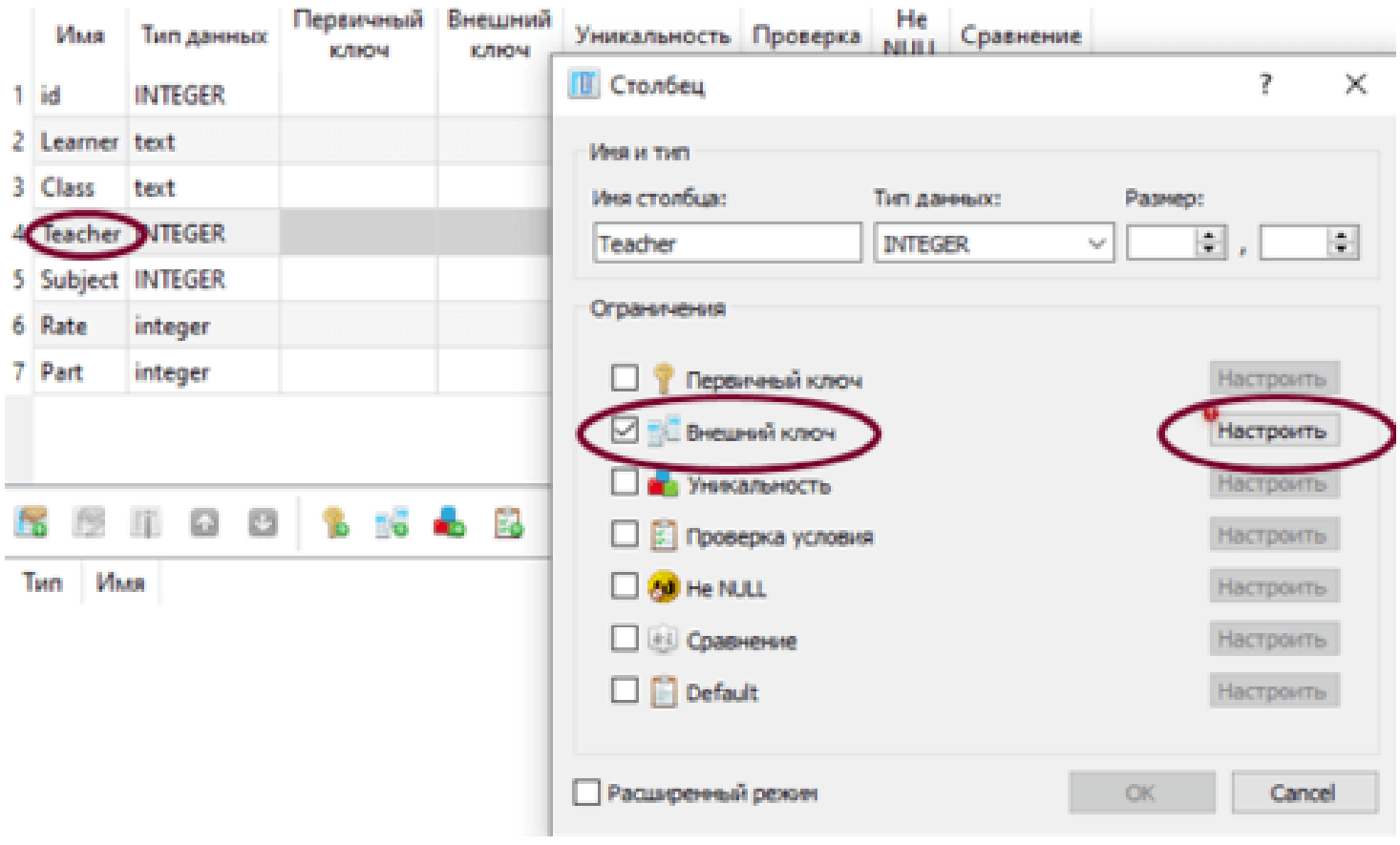
связь между их полями. В нашем случае нам необходимо связать между собой таблицы **Ученики** и таблицу **Учителя**, установив связь между полем **Учитель** в таблице **Ученики** и полем **id** из таблицы **Учителя.**



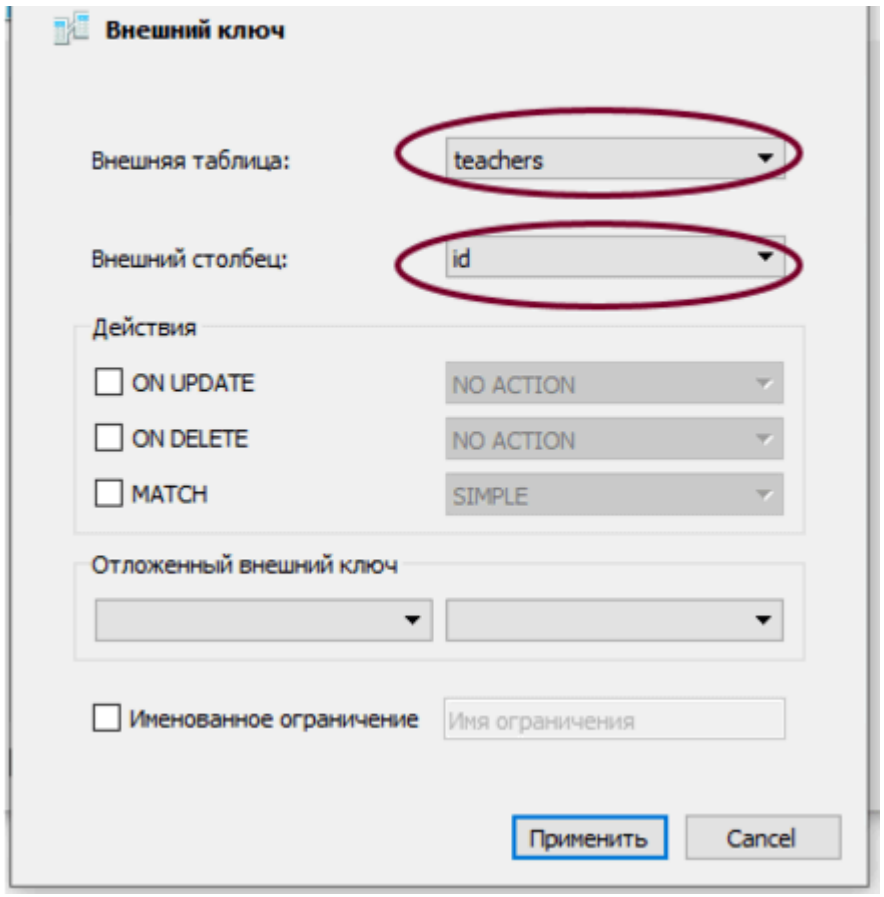

В таком случае поле **Учитель** в таблице **Ученики** будет являться **Внешним ключом**.
Это можно сделать с помощью SQL запроса, а можно и в SQLiteStudio. Сделаем это сначала в SQLiteStudio, заодно посмотрим, как будет выглядеть SQL запрос.
Сначала открываем структуру таблицы:




Щелкаем два раза по полю **Teacher** (Учитель) и в открывшемся окне ставим галочку **Внешний ключ**




И нажимаем кнопку **Настроить**.
Далее выбираем внешнюю таблицу и столбец в этой таблице, с которым будет связан столбец **Teacher** в таблице **learns**.




! Проблемная задача:
Теперь попробуйте в **SQLiteStudio** добавить в таблицу **learns** (Ученики) строку (запись) с новым учеником, у которого в поле **Teacher** (Учитель) будет стоять число, которого нет в столбце **id** таблицы **teachers**.
Объяснить результат.

Решение:
Возникнет ошибка.

! Проблемная задача:
Связать между собой таблицы учеников и предметов

# Урок 9. INNER JOIN.

На прошлом уроке мы установили связь между таблицами учеников, учителей и предметов.

**20 мин**

Но у нас есть кое-какая нерешенная проблема. При запросе записей из таблицы **learns** (Ученики) в полях Teacher (учитель) и Subject (предмет) стоят числа, а не имена учителей и названия предметов.
Иначе говоря, у нас все данные разделены по разным таблица и нам теперь необходимо каким-то образом получать данные из нескольких таблиц.
В этом нам поможет специальный оператор языка JOIN языка запросов SQL.
Как он работает?
Рассмотрим на нашем примере. Нам нужно получить все записи из таблицы **learns** и сделать так, чтобы вместо чисел в поле **Teacher** стояли имена учителей из таблицы **teachers**.
SQL запрос в таком случае будет выглядеть так:

```
SELECT * FROM learns INNER JOIN teachers ON learns.Teacher = teachers.id
```

Как видно начало запроса такое, как и раньше, мы выбираем записи из таблицы learns, но далее мы с помощью оператора INNER JOIN указываем, что необходимо добавить данные из таблицы teachers. И в конце после оператора ON указываем, что мы их связываем через поля **learns.Teacher** и **teachers.id.**

**!** Проблемная задача:
Реализовать в программе на Python данный запрос, вывести результат в виджет ListBox.

Результат, который мы получим будет выглядеть так:

| | | | | мет | | рть | | |
|---|---|---|---|---|---|---|---|---|
| 1 | Иванов Иван | 4а | 1 | 1 | 4 | 1 | 1 | Приставкина Е.У. |
| 2 | Петров | 6в | 2 | 2 | 4 | 1 | | |
| 3 | Кукушкина Лида | 7а | 10 | 3 | 5 | 1 | | |
| 4 | Варшавин Вова | 7а | 10 | 3 | 5 | 1 | | |
| 5 | Олегов Олег | 3г | 1 | 2 | 3 | 2 | | |
| 7 | Абовян Рашид | 5г | 3 | 3 | 5 | 3 | | |
| 8 | Яшин Вячеслав | 7е | 4 | 4 | 5 | 3 | | |

РЕЗУЛЬТАТ ЗАПРОСА

| id | Ученик | Класс | Учителя | Предмет | Оценки | Четверть | id | Учитель |
|---|---|---|---|---|---|---|---|---|
| 1 | Иванов Иван | 4а | 1 | 1 | 4 | 1 | 1 | Приставкина Е.У. |
| 5 | Олегов Олег | 3г | 1 | 2 | 3 | 2 | 1 | Приставкина Е.У. |

Оператор INNER JOIN взял только те записи из обеих таблиц, у которых совпадают между собой поля **learns.Teacher** и **teachers.id.**
И еще мы видим, что запрос нам вернул все поля из обеих таблиц.
Если нам нужны не все поля, то их можно указать после оператора SELECT:
SELECT Learner, Class, Subject, teachers.Teacher
FROM learns
INNER JOIN teachers ON learns.Teacher = teachers.id

Выполним этот запрос и увидим, что у нас вывелись только Имя ученика, Класс, Норме предмета и Имя учителя.
Заметим, что для поля **Teacher** из таблицы teachers нам необходимо указывать через точку название таблицы, так как после оператора **FROM** указано имя таблицы **learns**.

! Проблемная задача

Добавить в нашу программу на Python возможность формирования отчетов по ученикам, учителям и предметам с возможностью выбора выводимых полей (столбцов). В столбцах Учитель, Предмет и Ученик должны быть имена и названия (а не числа **id**).

# Урок 10. LEFT JOIN. RIGHT JOIN.

На прошлом уроке мы изучили оператор INNER JOIN, которые позволяет получить данные записей из разных таблиц, у которых совпадают два поля.

Бывают случаи, когда нам нужно получить все записи из какой-либо таблицы и еще записи из второй таблицы, у которой какое-нибудь поле совпадает с полем из первой таблицы.
Либо наоборот, получить записи из таблицы, у которых указанное поле совпадает с полем из второй таблицы и все записи из второй таблицы.
Для этого соответственно используются операторы:
LEFT JOIN
RIGHT JOIN

! Проблемная задача:
Проверить работу этих операторов с помощью, соответствующих SQL запросов.

! Проблемная задача:
Составить SQL запрос, который выводит названия всех предметов и учеников, которые изучают соответствующий предмет. Если в базе данных есть предмет, по которому нет ни одного ученика, этот предмет все равно должен быть в отчете.

! Проблемная задача:
Составить SQL запрос и вывести на экран следующий отчет: Наименования всех классов и список учеников по каждому классу. Если в базе данных есть класс, в котором пока нет ни одного ученика, этот класс все равно должен быть в отчете.

# Урок 11. Типы связей.

Если посмотреть на нашу таблицу с учениками, то можно заметить, что классы нужно тоже вынести в отдельную таблицу, также как мы создали отдельные таблицы для учителей и предметов.

Сделаем это:

| | | | | | | |
|---|---|---|---|---|---|---|
| 1 | Иванов Иван | 4 | 1 | 1 | 4 | 1 |
| 2 | Петров | 6 | 2 | 2 | 4 | 1 |
| 3 | Кукушкина Лида | 8 | 10 | 3 | 5 | 1 |
| 4 | Варшавин Вова | 8 | 10 | 3 | 5 | 1 |
| 5 | Олегов Олег | 3 | 1 | 2 | 3 | 2 |
| 7 | Абовян Рашид | 5 | 3 | 3 | 5 | 3 |
| 8 | Яшин Вячеслав | 9 | 4 | 4 | 5 | 3 |

| | |
|---|---|
| 1 | Приставкина Е.У. |
| 2 | Циркулев А.В. |
| 3 | Природин Т.Р |
| 4 | Сажин Т.Л |
| 5 | Теплов Л.И. |
| 10 | Лыжин Т.П. |

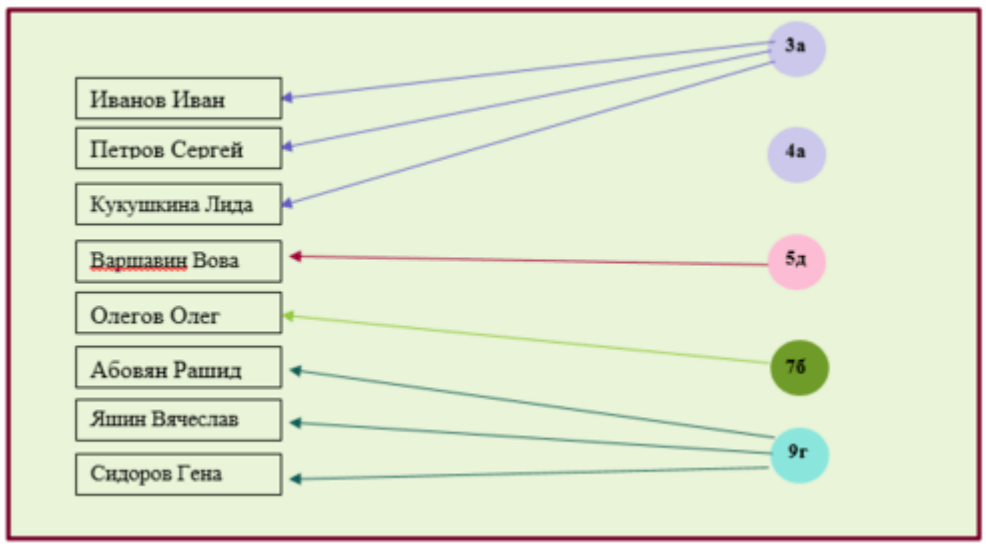
**subjects**

| id | Subject |
|---|---|
| 1 | Русский |
| 2 | Математика |
| 3 | Физкультура |
| 4 | География |

**classes**

| id | Class |
|---|---|
| 1 | 1б |
| 2 | 3а |
| 3 | 3г |
| 4 | 4а |
| 5 | 5г |
| 6 | 6в |
| 7 | 6б |
| 8 | 7а |
| 9 | 7е |

Можно сказать, что, если мы возьмем наугад любого ученика, то ему будет соответствовать только один класс. Один ученик не может учиться сразу в двух и более классах. Если Иванов Иван учится в 7д, то уже не может учиться в каком-нибудь 7а или 9в.
А вот если мы возьмем какой-нибудь класс, например 7а, то мы видим, что ему соответствует 2 (два) ученика: Кукушкина Лида и Варшавин Вова. Да и в реальности вы все знаете, что одному классу соответствует несколько учеников, точнее все, кто учится в этом классе.
Получаем такую связь:




Такую связь называют:
**ОДИН КО МНОГИМ**

**!** Проблемная задача:
Привести 3 примера из реальной жизни связи ОДИН КО МНОГИМ.

! Проблемная задача:
Добавить в программу возможность выводить отчеты по двум полям из разных таблиц. Например, список учителей, а по каждому учителю список предметов, которые он ведет.

Наверное, вы можете догадываться, что раз есть связь ОДИН КО МНОГИМ, то значит должна существовать и связь МНОГИЕ КО МНОГИМ.

В нашей базе данных ШКОЛА уже есть несколько таких связей.
! Проблемная задача:
Определить все связи МНОГИ КО МНОГИМ в базе данных Школа.

# Урок 12. Типы связей.

У одного ученика имеется несколько учителей, а у одного учителя имеется несколько учеников. Такая связь называется МНОГИЕ СО МНОГИМИ. Наличие между таблицами связи МНОГИЕ СО МНОГИМИ считается не правильным.
Почему?
Посмотрим на БД Школа:

| | | | | | | |
|---|---|---|---|---|---|---|
| 1 | Иванов Иван | 4 | 1 | 1 | 4 | 1 |
| 2 | Петров | 6 | 2 | 2 | 4 | 1 |
| 3 | Кукушкина Лида | 8 | 10 | 3 | 5 | 1 |
| 4 | Варшавин Вова | 8 | 10 | 3 | 5 | 1 |
| 5 | Олегов Олег | 3 | 1 | 2 | 3 | 2 |
| 7 | Абовян Рашид | 5 | 3 | 3 | 5 | 3 |
| 8 | Яшин Вячеслав | 9 | 4 | 4 | 5 | 3 |

| | |
|---|---|
| 1 | Приставкина Е.У. |
| 2 | Циркулев А.В. |
| 3 | Природин Т.Р |
| 4 | Сажин Т.Л |
| 5 | Теплов Л.И. |
| 10 | Лыжин Т.П. |

**subjects**

| id | Subject |
|---|---|
| 1 | Русский |
| 2 | Математика |
| 3 | Физкультура |
| 4 | География |

**classes**

| id | Class |
|---|---|
| 1 | 1б |
| 2 | 3а |
| 3 | 3г |
| 4 | 4а |
| 5 | 5г |
| 6 | 6в |
| 7 | 6б |
| 8 | 7а |
| 9 | 7е |

Иванов Иван по Русскому имеет оценку 4 в 1 четверти, преподаватель у него по этому предмету Приставкина Е.У.
Добавим в таблицу запись о других предметах, например о математике:

**learns**

| id | Learner | ClassID | TeachID | SubID | Rate | Part |
|---|---|---|---|---|---|---|
| 1 | Иванов Иван | 4 | 1 | 1 | 4 | 1 |
| 2 | Петров | 6 | 2 | 2 | 4 | 1 |
| 3 | Кукушкина Лида | 8 | 10 | 3 | 5 | 1 |
| 4 | Варшавин Вова | 8 | 10 | 3 | 5 | 1 |
| 5 | Олегов Олег | 3 | 1 | 2 | 3 | 2 |
| 7 | Абовян Рашид | 5 | 3 | 3 | 5 | 3 |
| 8 | Яшин Вячеслав | 9 | 4 | 4 | 5 | 3 |
| 9 | Иванов Иван | 4 | 2 | 2 | 5 | 1 |

**teachers**

| id | Teacher |
|---|---|
| 1 | Приставкина Е.У. |
| 2 | Циркулев А.В. |
| 3 | Природин Т.Р |
| 4 | Сажин Т.Л |
| 5 | Теплов Л.И. |
| 10 | Лыжин Т.П. |

**subjects**

| id | Subject |
|---|---|
| 1 | Русский |
| 2 | Математика |
| 3 | Физкультура |
| 4 | География |

**classes**

| id | Class |
|---|---|
| 1 | 1б |
| 2 | 3а |
| 3 | 3г |
| 4 | 4а |
| 5 | 5г |
| 6 | 6в |
| 7 | 6б |
| 8 | 7а |
| 9 | 7е |

Мы видим, что Иванов Иван уже дважды присутствует в таблице учеников. У них разный **id** так-как это первичный ключ. Хотя человек это один. И еще, с точки зрения оптимального расходования памяти, такой вариант не самый лучший. В таблице учеников должны быть ученики и записаны они там должны быть по одному разу, как в школьном журнале.
В таком случае, как реализовать связь МНОГИЕ КО МНОГИМ?

! Проблемная задача.
Попробовать придумать решение проблемы.

Решение заключается в том, что мы создаем еще одну таблицу – таблицу связей учеников и учителей.
Назовем ее **learntoteach**.
**В ней будет поле LearnID, указывающее на id в таблице учеников и поле TeachID, указывающее на id таблице учителей.**

| | | | | | |
|---|---|---|---|---|---|
| 1 | Иванов Иван | 4 | 1 | 4 | 1 |
| 2 | Петров | 6 | 2 | 4 | 1 |
| 3 | Кукушкина Лида | 8 | 3 | 5 | 1 |
| 4 | Варшавин Вова | 8 | 3 | 5 | 1 |
| 5 | Олегов Олег | 3 | 2 | 3 | 2 |
| 7 | Абовян Рашид | 5 | 3 | 5 | 3 |
| 8 | Яшин Вячеслав | 9 | 4 | 5 | 3 |

| id | Class |
|---|---|
| 1 | Приставкина Е.У. |
| 2 | Циркулев А.В. |
| 3 | Природин Т.Р |
| 4 | Сажин Т.Л |
| 5 | Теплов Л.И. |
| 10 | Лыжин Т.П. |

**learntoteach**

| LearnID | TeachID |
|---|---|
| 1 | 1 |
| 1 | 2 |
| 2 | 2 |
| 3 | 10 |
| 4 | 10 |
| 5 | 2 |
| 7 | 10 |
| 8 | 4 |

**classes**

| id | Class |
|---|---|

Теперь у нас реализована связь М-М с помощью вспомогательной таблицы learntoteach. В этой таблицы по сути перечислены пары id ученика и id учителя.
Заметим, что поле id мы не создали в этой таблице. Потому что пара полей LearnID и TeachID является составным ключом. В этой таблице не может быть двух записей с одинаковой парой полей LearnID и TeachID.

! Проблемная задача:
По аналогии устранить оставшиеся связи М-М в таблицах, путем создания вспомогательной таблицы.

**! Модифицировать SQL запросы для получения данных с учетом сделанных изменений в БД (убрали связь М-М, добавили новую таблицу).**

# Урок 13. Сортировка. Группировка. Функции.

На этом уроке мы узнаем, как можно упорядочить или отсортировать получаемые записи из таблицы, как их сгруппировать и познакомимся с агрегирующими функциями

Предположим, нам необходимо вывести список всех учеников, отсортированных по возрастанию класса.
Делается это с помощью оператора OREDER BY.
Пример:

```
SELECT Learner, Class
FROM learners
ORDER BY Class
```

Результат:

```
Олегов Олег 3г
Иванов Иван 4а
Абовян Рашид 5г
Петров 6в
Кукушкина Лида 7а
Варшавин Вова 7а
Яшин Вячеслав 7е
```

Если необходимо отсортировать по убыванию, то добавляем слово DESC в конец строки с оператором ORDER BY

```
SELECT Learner, Class
FROM learners
ORDER BY Class DESC
```

Если поле, по которому идет сортировка будет текстовым, то сортировка будет в алфавитном порядке.

**! Проблемная задача:**
Составить SQL запрос, который выводит список имен всех учителей в обратном алфавитном порядке.

! Проблемная задача:
Добавить в программу на python возможность пользователю выбирать по какому полю сортировать записи. Можно использовать Radiobutton.

Представим, что нам нужно посчитать сумму баллов ученика Иванова Ивана за все время.

! Проблемная задача:
Составить SQL запрос для вывода всех оценок Иванова Ивана.

Решение:

```
SELECT Learner, Rate, Part
```

ON learners.id = estimations.LearnerID

Выполним этот запрос и получим таблицу с оценками.
Получается, что нам нужно посчитать сумму всех оценок в столбце Rate.
Для этого необходимо в строке с оператором **SELECT** к полю **Rate** применить агрегирующую функцию **sum**():

SELECT Learner, sum(Rate)
FROM learners
INNER JOIN estimations
**ON learners.id = estimations.LearnerID**
Выполним этот запрос:

| | Learner | sum(Rate) | Part |
|---|---|---|---|
| 1 | Иванов Иван | 12 | 2 |

Получили сумму всех оценок Иванова Ивана.

! Проблемная задача:
Посчитать средний бал любого ученика в 1 четверти.

В начале урока мы говорили еще про группировку данных.
Если нам нужно посчитать количество учеников в каждом классе и вывести эту информацию в виде отчета – список классов с количеством учеников, то для этого можно использовать оператор GROUP BY.

Пример:
SELECT Class, count(Class)
FROM classes
INNER JOIN learners
ON learners.ClassID = classes.id
GROUP BY Class

Результат:

| | Class | count(Class) |
|---|---|---|
| 1 | 3г | 1 |
| 2 | 4а | 3 |
| 3 | 5г | 2 |
| 4 | 6в | 1 |
| 5 | 7а | 3 |
| 6 | 7е | 1 |

! Проблемная задача:
Вывести отчет по любому ученику, средний бал по каждому предмету.

! Проблемная задача:
Вывести отчет по количеству учеников у каждого учителя.

# Урок 14. Табличный виджет.

При работе программы или приложения с базой данных было бы удобно выводить данные в форме таблицы.
В библиотеке tkinter есть такой виджет. Он называется treeview b находится он в модуле ttk.
Импортируем этот модуль:
import tkinter.ttk as ttk
from tkinter import *

Создадим объект класса Treeview:

Ttable = ttk.Treeview(window)

Чтобы задать столбики используется такая запись:
Ttable["columns"]=("one","two","three")

Либо такая:

Либо при создании виджета:
Ttable = ttk.Treeview(window, column=(1,2,3))

Таким образом мы указываем сколько будет столбцов и задаем им индексы.
Далее, используя эти индексы мы можем задавать параметры столбцам.
Зададим ширину и заголовки:
Ttable["columns"]=(1,2,3)
Ttable.column("#0", width=50)
Ttable.column(1, width=250)
Ttable.column(2, width=150)
Ttable.column(3, width=80)
Ttable.heading(1, text="Ученик")
Ttable.heading(2, text="Класс")
Ttable.heading(3, text="Оценка")

У нас есть одна запись:
Ttable.column("#0", width=50)

Дело в том, что у виджета treeview по умолчанию всегда есть один нулевой столбик, а индекс этого столбца равен **“#0”.**

Чтобы вставить строку в виджет используется метод **insert**.

Ttable.insert('',0,text='1', values=('Иванов Иван', '4a', 5))

Первый параметр – пустая строка, о нем мы поговорим дальше.
Второй параметр – номер строки, после которой будет вставлена новая строка. **0** – строка будет вставляться в начало, ‘**end’** – строка будет вставляться в конец.
values – задает остальные значения столбцов по порядку.

! Проблемная задача:
Модифицировать программу, заменив виджет Listbox на Treeview.

# Урок 15. Контрольный урок.

! Проблемная задача:
Спроектировать БД для чемпионата по игре в ConterStrike.
БД должна содержать следующие данные:
Имена игроков.
Названия команд.
Информацию о том, какой игрок в какой команде.
Информацию о сыгранных играх: какая команда с какой играла, кто выиграл, с каким счетом.
Написать программу на python для работы с этой БД.
Программа должна выполнять следующие задачи:
Добавление, редактирование и удаление команд и игроков.
Занесение и удаление информации об сыгранных играх.
Вывод отчетов по игрокам, командам и играм.

Также на данном уроку ученикам предлагается выбрать итоговый проект.
Варианты итогового проекта:
- игра крестики нолики с БД игроков и их результатах.
- игра морской бой нолики с БД игроков и их результатах.
- справочник домашних животных
- программа по управлению магазином
- программа Экзаменатор, которая тестирует учеников по предметам и хранит данные в БД.
- Электронный фотоальбом. (БД хранящая фотографии, и выводящая отчет по фото в разрезе темы или временного промежутка)

# Урок 16. Защита проектов.

**Предлагаемый сценарий проведения урока:**
- Жеребьевка очередности учеников

| **10 мин** |
| --- |

- Рассказ ученика о своей программе, презентация ее работы, рассказ об используемых алгоритмах, виджетах

**15 мин / 1 ученик**

- Вопросы ученику

**5 мин / 1 ученик**

*Примеры вопросов:*

- Сколько используется таблиц в БД
- Какие типы связей используются между таблицами
- В каких таблицах используется составной первичный ключ и почему
- Какие виджеты используются в интерфейсе для отображения данных
- Используются ли в программе SQL запросы с операторами INNER JOIN, LEFT JOIN или RIGHT JOIN?
- Если используется оператор JOIN, то для вывода каких данных, из каких таблиц.
- Приводили ли вы БД к нормальным формам, к каким?
- После получения данных по SQL запросу вы отображаете их сразу как есть или применяете к ним какой-то алгоритм обработки, если да, то какой?
- Что подтолкнуло вас на создание такого приложения?

## Критерии оценки проектов

1. БД приведена к 1 нормальной форме
2. БД приведена ко 2 нормальной форме
3. БД приведена к 3 нормальной форме
4. В БД нет связи МНОГИЕ КО МНОГИМ
5. В программе присутствует дополнительная обработка данных получаемых по SQL запросу. (например, после получения данных в список кортежей, происходит какой-либо поиск по данным или выборка по какому-то критерию, либо математическая обработка, например, подсчет среднего арифметического)
6. Дружелюбный интерфейс (понятное меню и кнопки)

Используем cookies для корректной работы сайта, персонализации пользователей и других целей, предусмотренных политикой обработки персональных данных.