* [ОБЗОР КУРСА](https://lyceum.yandex.ru/courses/165/groups/1257)

[Урок QT SQL 1](https://lyceum.yandex.ru/courses/165/groups/1257/lessons/1230)

**Введение в БД, работа с SQL-таблицами и отображение данных в PyQT. Часть 1**

1. [Введение в базы данных](https://lyceum.yandex.ru/courses/165/groups/1257/lessons/1230/materials/2565#1)
2. [Основы SQL](https://lyceum.yandex.ru/courses/165/groups/1257/lessons/1230/materials/2565#2)
3. [SQL + Python](https://lyceum.yandex.ru/courses/165/groups/1257/lessons/1230/materials/2565#3)

**Аннотация**

*На уроке мы познакомимся с базами данных и языком SQL.*

**Введение в базы данных**

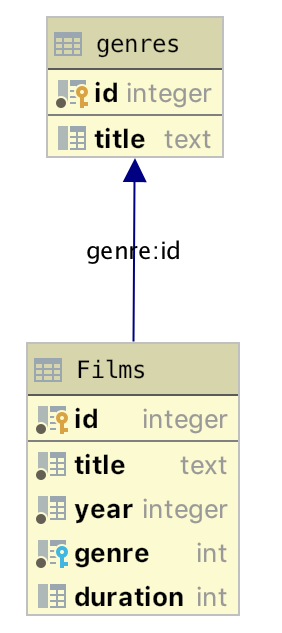
На прошлых уроках вы уже сталкивались с хранением данных во внешних источниках: простых текстовых документах, документах с особым форматированием, например, csv-таблицах. Но зачастую такие хранилища данных неудобны, потому что дублируются несколько раз, занимают место на жестком диске, а поиск в них занимает много времени. Для решения этих проблем был придуман такой способ хранения информации, как база данных (БД).

Существуют различные системы управления базами данных, так называемые СУБД. База данных — непосредственное хранилище информации, а СУБД предоставляют разработчикам интерфейс для общения с базой. В наших проектах мы будем использовать СУБД **SQLite**. Основным достоинством этой СУБД является возможность упаковки всей базы в один файл и встроенная в Python по умолчанию библиотека для работы с ней.

Данные в БД хранятся с помощью таблиц и связей между этими таблицами. Мы будем работать с БД, в которой хранится информация о фильмах. Их название, год выпуска, жанр и продолжительность в минутах. Каждое поле занимает определенный объем, в зависимости от типа данных: целочисленный, строковый или используемый для хранения времени.

Если бы мы хранили информацию, относящуюся к каждому фильму, нам для каждой записи приходилось бы занимать место для названия жанра. Но, поскольку число жанров ограничено и они наверняка будут повторяться для разных фильмов, мы можем создать дополнительную таблицу, в которой будем хранить пару формата **Ключ: Значение**. В качестве ключа используется уникальный идентификатор (id), а в качестве значения — название жанра. А в первой таблице просто хранить ссылку на вторую.

Очень часто для наглядного представления таблиц и связей используют такую визуализацию:



Именно так выглядит база данных, с которой мы будем работать. Скачать ее можно [здесь](https://yastatic.net/s3/lyceum/content/images/second-year/pyqt-6/films.db).

**Основы SQL**

Для работы с базами данных был придуман специальный язык — SQL (structured query language — «язык структурированных запросов»). Основной командой для получения какой-либо информации из БД является команда SELECT. Ее синтаксис выглядит так:

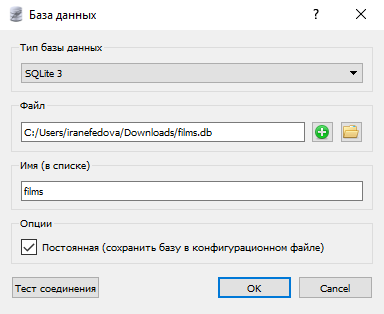
SELECT ЧТО FROM ИМЯ\_ТАБЛИЦЫ

WHERE УСЛОВИЕ

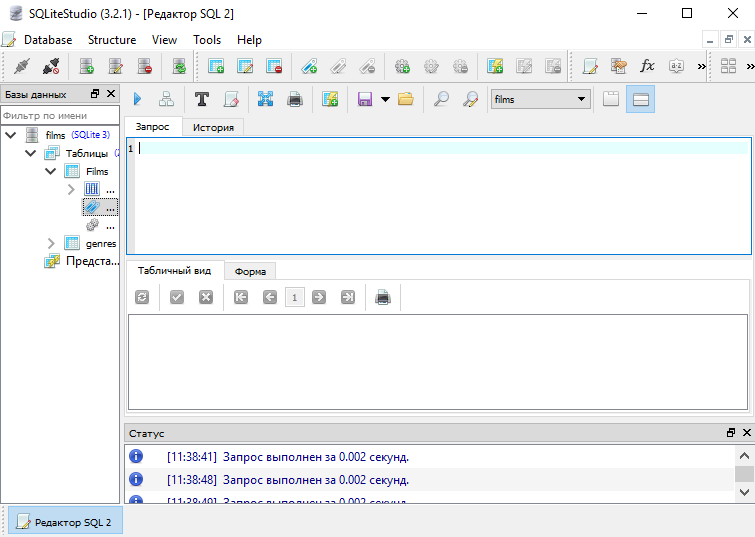
Кроме этого, есть и различные модификаторы этой команды. Например, ORDER BY ПОЛЕ — тогда результаты будут выведены в отсортированном виде по заданному полю. В условии может быть и вложенный запрос.

Перед тем как начать работать с БД с помощью Python, поработаем с ней с помощью официальной программы [SQLiteStudio](https://sqlitestudio.pl/). Первое, что необходимо сделать после установки, — добавить нашу базу фильмов.

Для этого в основном меню необходимо выбрать пункт Add a Database и в открывшемся окне указать путь к файлу нашей БД.



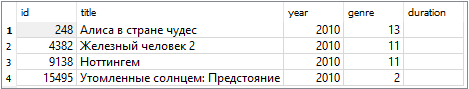
Если все прошло успешно, можно писать запросы. Для этого необходимо открыть редактор SQL. Его логотип выглядит как свиток бумаги с карандашом. Интерфейс редактора состоит из двух частей: первая, где пишется сам запрос, и вторая, где отображаются результаты.



Напишем наш первый запрос. Получим все фильмы, выпущенные в 2010 году.

SELECT \* FROM Films

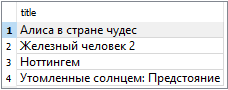
WHERE year = 2010



Символ \* обозначает, что нам необходимо получить все поля. Однако очень часто нам нужно получить только одно или два поля. Модифицируем запрос так, чтобы выводилось только название.

SELECT title FROM Films

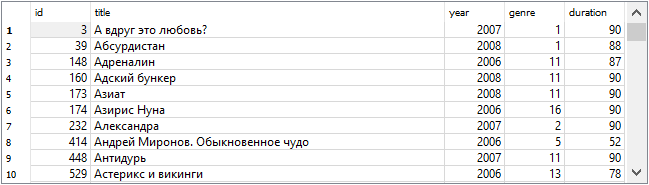
WHERE year = 2010



Условий может быть и несколько: выберем фильмы, выпущенные после 2005 года с продолжительностью от 40 минут до 1,5 часов:

SELECT \* FROM Films

WHERE (year > 2005) AND (duration >= 45) AND (duration <= 90)



А как вывести все фильмы определенного жанра, например, фантастику?

SELECT title FROM Films

WHERE genre=(

SELECT id FROM genres

WHERE title = 'фантастика')

Сначала выполнится внутренний запрос: из таблицы genres будет получен id категории «Фантастика», а затем будет выполнено сравнение и выведен результат.

Помимо того, может быть выполнено сравнение не с одним элементом, а проверка на попадание в список. Это делается с помощью специального слова IN. Например:

SELECT title, duration FROM Films

WHERE duration IN (45,90)

Кроме строго сравнения (больше, меньше, равно), существуют специальные операторы, облегчающие жизнь. В этом уроке мы рассмотрим два.

Первый — BETWEEN — проверяет, попадает ли заданное значение в диапазон.

SELECT \* FROM Films

WHERE (year > 2005) AND duration BETWEEN 45 AND 60

И второй — оператор LIKE, с помощью которого можно проверить, насколько похожа та или иная строка на заданный шаблон. Для шаблона используются специальные символы:

* % — обозначает любое количество, в том числе нулевое, любых символов
* \_ — обозначает один любой символ

Рассмотрим пример с нашей базой. Получим список фильмов, у которых первая буква в названии — **А** и третья — **к**.

SELECT \* FROM Films

WHERE title like 'А\_к%'



Из любого условия достаточно просто сделать обратное. Для этого перед ним необходимо добавить специальное слово — NOT.

Кроме этого, есть возможность избавиться от повторов строк, используя в запросе специальное слово — DISTINCT. Например, вот так можно получить список по годам, в которые выходили фильмы в нашей базе данных, без повторений.

SELECT DISTINCT year FROM Films

**SQL + Python**

Для работы с SQLite из Python существует специальная библиотека — sqlite3. Очень часто подход при работе с БД из различных языков программирования стандартизирован. Существуют два основных понятия: подключения и курсоры.

* **Подключение** — объект, в котором чаще всего указывается либо путь к файлу, либо путь к серверу. Он отвечает только за подключение к БД и, соответственно, отключение от нее
* **Курсор** — объект, в котором непосредственно и производится работа с БД

Напишем программу (пока что без графического интерфейса), которая получает результаты одного из рассмотренных выше запросов и выводит их в консоль.

# Импорт библиотеки

import sqlite3

# Подключение к БД

con = sqlite3.connect("films.db")

# Создание курсора

cur = con.cursor()

# Выполнение запроса и получение всех результатов

result = cur.execute("""SELECT \* FROM Films

WHERE year = 2010""").fetchall()

# Вывод результатов на экран

for elem in result:

print(elem)

con.close()

(248, 'Алиса в стране чудес', 2010, 13, '')

(4382, 'Железный человек 2', 2010, 11, '')

(9138, 'Ноттингем', 2010, 11, '')

(15495, 'Утомленные солнцем: Предстояние', 2010, 2, '')

Как можно заметить, результат запроса — это кортеж кортежей.

Метод .fetchall() возвращает все полученные элементы. Существует еще метод .fetchone(), возвращающий, как несложно догадаться, только первый элемент, и метод .fetchmany(n), возвращающий n первых записей.

Для запросов очень часто необходимо указывать какие-либо параметры: год, продолжительность и т. д. Для этого существует удобный синтаксис. Вместо значения в запросе указывается вопросительный знак, а затем в итерируемом объекте (чаще всего в кортеже) указываются параметры.

result = cur.execute("""SELECT \* FROM Films

WHERE year = ? and duration > ?""",(2010, 90)).fetchall()

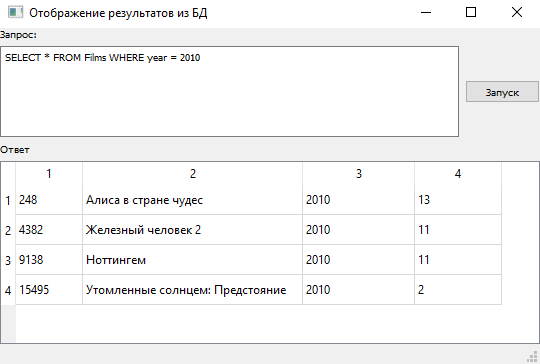
Важно не забыть, что, если мы указываем кортеж из одного элемента, нам все равно необходимо после него поставить запятую.

result = cur.execute("""SELECT \* FROM Films

WHERE year = ?""",(2009,)).fetchall()

Но не будем забывать и о графическом интерфейсе. Напишем программу, которая будет отображать результаты введенного запроса в таблице.

С помощью QtDesigner создадим интерфейс: поле для ввода запроса, таблица для отображения результатов и кнопка для запуска выполнения запроса. Работа с **QTableWidget** разбиралась в прошлом уроке, когда мы работали с csv-файлами. В данном случае принцип абсолютно такой же.



В этом уроке мы изучили основной запрос языка SQL, научились работать с БД с помощью Python. Больше узнать про синтаксис команды SELECT можно [здесь](http://www.sqlitetutorial.net/).

[Справка](https://yandex.ru/support/lyceum-students)

Исключительное право на учебную программу и все сопутствующие ей учебные материалы, доступные в рамках проекта «Яндекс.Лицей», принадлежат АНО ДПО «ШАД». Воспроизведение, копирование, распространение и иное использование программы и материалов допустимо только с предварительного письменного согласия АНО ДПО «ШАД».

© 2018 – 2020  ООО «[Яндекс](https://yandex.ru/)»

Чаты