**Задание проекта**

(его следует согласовать с преподавателем до начала работы)

Для решения задач в Яндек.Лицее на уроках по изучению баз данных и работе с SQL-таблицами необходимо было строить SQL запросы.

Чтобы понимать, как построить запрос необходимо знать наименование таблиц, наименование полей таблиц, знать какие данные содержит таблица. А чтобы проверить правильно ли построен запрос, необходимо иметь понятие о том сколько и каких записей должно быть отобрано.

Задание проекта: написать программу для просмотра произвольной базы данных SQLite.

**Пояснительная записка**

(документ, описывающий суть проекта, его особенности и прочие моменты)

Для решения задач в Яндек.Лицее на уроках по изучению баз данных и работе с SQL-таблицами необходимо было строить SQL запросы.

Чтобы понимать, как построить запрос необходимо знать наименование таблиц, наименование полей таблиц, знать какие данные содержит таблица. А чтобы проверить правильно ли построен запрос, необходимо иметь понятие о том сколько и каких записей должно быть отобрано.

Цель проекта: написать программу для просмотра произвольной базы данных SQLite.

Задачи (условия):

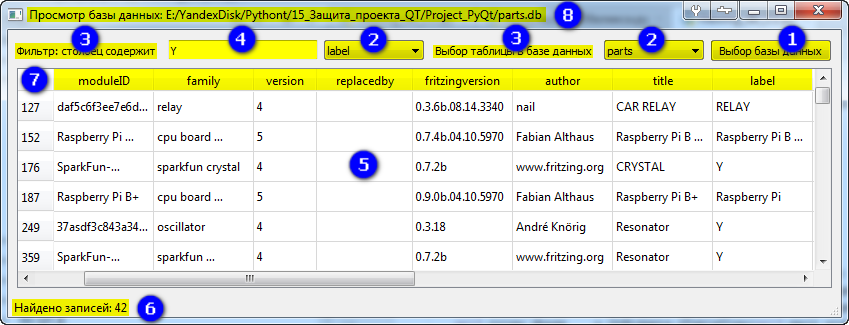
* возможность просмотра произвольной базы данных SQLite;
* вывод наименований таблиц;
* вывод наименований полей;
* вывод количества записей в таблице;
* фильтрация данных по произвольному полю с выводом количества записей.

Проект состоит из одного окна, на котором располагается 7 элементов:

1. 1 QPushButton для вызова диалогового окна выбора базы данных
2. 2 QComboBox для выбора таблиц и выбора полей таблиц
3. 2 QLabel – пояснительные подписи к активным элементов
4. 1 QLineEdit – используется для фильтрации
5. 1 QTableView – для просмотра данных

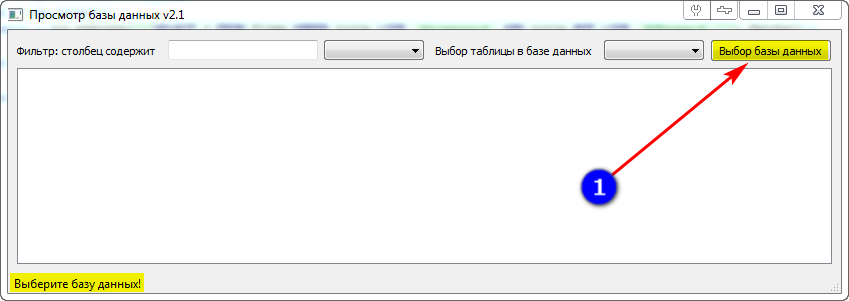
Кроме этого задействованы

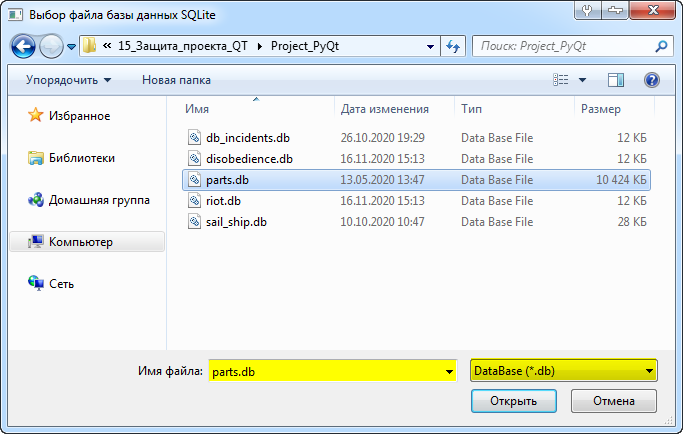
1. строка состояния (statusBar) для вывода дополнительной информации.
2. заголовки полей таблицы (horizontalHeader QTableView) для формирования «контекстного меню».
3. заголовок окна (WindowTitle), в котором выводится информация о расположении просматриваемой базы данных.



**Работа с программой**

При работе с программой на первом шаге необходимо выбрать интересующую базу данных SQLite.

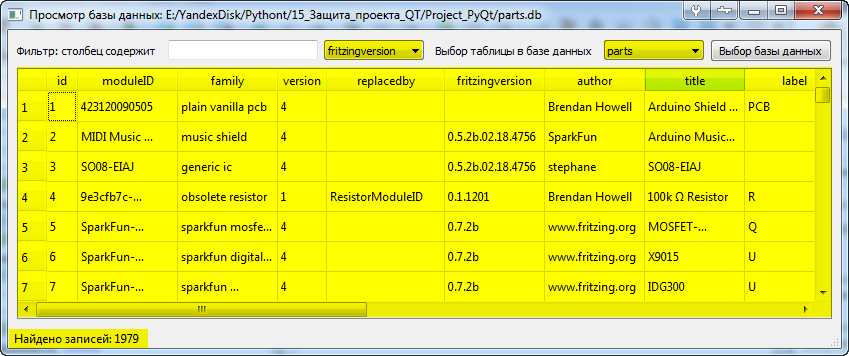




Программа протестирована на файлах с расширением sqlite, db и db3

После выбора базы данных:

* QComboBox заполняется списком таблиц созданных в базе данных
* первая по порядку таблица (активная в QComboBox) открывается в QTableView
* второй QComboBox заполняется списком полей открытой таблицы
* в statusBar отображается количество записей.



QComboBox со списком таблиц предназначен для выбора отображаемой таблицы.

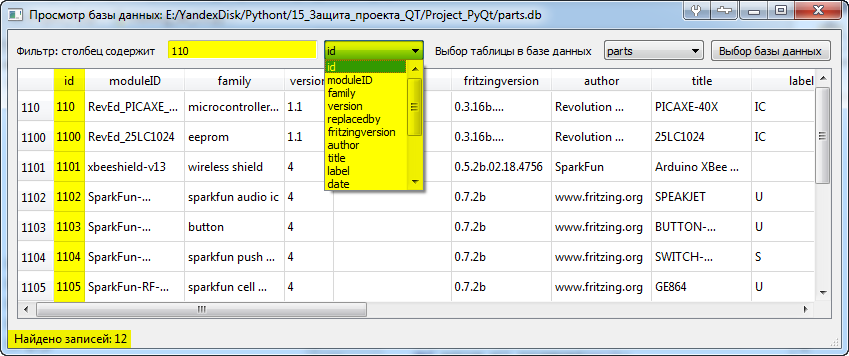
Но только одного представления таблиц недостаточно чтобы помочь в построении правильного запроса. Часто необходимо знать сколько и каких значений содержит то или иное поле. Это помогает понять логику построения запроса, а также проверить его правильность.

Поэтому большая часть кода предназначена для организации быстрой и наглядной фильтрации по таблице.

Условно (по действиям пользователя) можно отметить два типа фильтрации:

1. С использованием QLineEdit и QComboBox со списком полей открытой таблицы.
2. При помощи контекстного меню вызываемого по нажатию на заголовки полей таблицы (horizontalHeader QTableView).

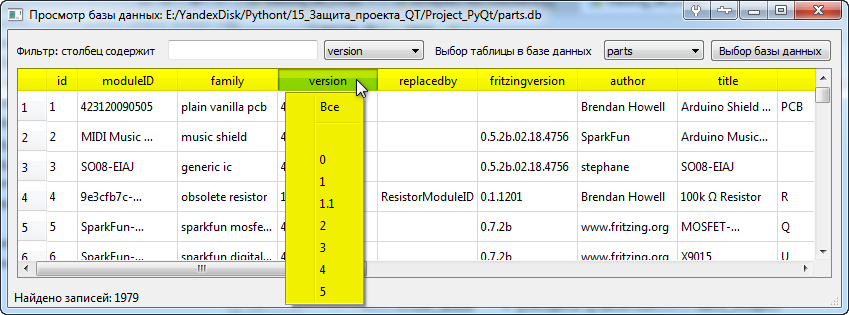
**Первый тип фильтрации**



Фильтр отбирает записи содержащие символы, введенные в QLineEdit по полю, выбранному QComboBox.

Фильтрация осуществляется «налету», сразу при вводе текста. При выборе другого поля в QComboBox, фильтрация сразу выполняется по нему. В statusBar выводится информация о количестве записей.

**Второй тип фильтрации.**



По клику мыши на заголовке поля таблицы (horizontalHeader QTableView), выбираются уникальные значения из которых формируется всплывающее меню. При клике на соответствующем значении, отфильтровываются записи, содержащие данное значение в выделенном поле.

Для порядка меняются значения QLineEdit и QComboBox используемых в первом типе фильтрации.

**Программный код**

(с необходимыми комментариями, пояснениями и ресурсами)

Программный код состоит из 9 модулей:

* модуль инициализации
* 4 модуля связанные с обработкой событий элементов формы
* 3 модуля с созданием и обработкой событий контекстного меню
* отдельно написан модуль для подсчета количества записей, по умолчанию в модель данных загружается только 256 записей.

**Модуль инициализации (def** \_\_init\_\_(self, parent=**None**))

В модуле инициализации элементы размещаются на gridLayout, задаются необходимые подписи и события.

Пришлось указать коэффициенты растяжения элементов верхней строки т.к. при загрузке данных QComboBox не изменяют свой размер.

**Событие обработки нажатия QPushButton (def** pushB\_dbase\_clicked(self))

В данном модуле осуществляется выбор файла базы данных SQLite. Для этого вызывается диалоговое окно выбора файла базы данных.

После выбора база данных загружается, а первый QComboBox заполняется списком таблиц. При заполнении списка вызывается событие QComboBox, в котором осуществляется вывод таблицы.

В Заголовке (WindowTitle) указывается полный путь к файлу базы данных, с которым осуществляется работа.

**Событие обработки выбора таблицы в QComboBox (def** combo\_dbase\_currentIndexChanged(self, Index))

В данном модуле создается модель данных на основе выбранной таблицы. На ее основе задается модель, обеспечивающая сортировку и фильтрацию данных (QSortFilterProxyModel).

Далее эта модель выводится в QTableView. Второй QComboBox заполняется наименованиями полей таблицы. Сбрасывается строка поиска QLineEdit

**Событие при изменении текста в QLineEdit (def** line\_search\_textChanged(self, text))

Основной модуль, в который выполняет фильтрацию данных. При каждом изменении QLineEdit выполняется поиск текста в заданном поле (столбце).

Здесь же проверятся наличие таблиц в базе данных, чтобы исключить ошибку. А после применения фильтра вызывается модуль подсчета записей.

**Событие обработки выбора столбца в QComboBox**

(**def** combo\_field\_currentIndexChanged(self, index))

В данном модуле у модели изменяется столбец (поле) по которому осуществляется фильтрация данных. Фильтрация выполняется немедленно.

Вызывается модуль подсчета записей.

**Событие обработки нажатия на заголовки столбцов в QTableView**

(**def** tview\_dbase\_horizontalHeader\_sectionClicked(self, logicalIndex))

В этом модуль формируется контекстное меню (QMenu) с уникальными значениями по клику на заголовке столбца QTableView.

Модель данных загружается до конца, выбираются уникальные значения. В меню добавляются элементы с выбранными уникальными значениями, а также элемент «все». Каждому задается событие, вызываемый по клику на пункте меню.

Задается позиция для вывода меню.

Для QComboBox задается значение в соответствии с выбранным столбцом.

**Событие при выборе элемента (QAction) «все» в контекстном меню (QMenu)**

**(def** action\_all\_triggered(self))

Для QLineEdit задается пустое текстовое значение, что вызывает фильтрацию.

**Событие при выборе уникального элемента (QAction) в контекстном меню (QMenu)**

**(def** signalMapper\_mapped(self, i))

Для QLineEdit задается текстовое значение в соответствии с наименованием, что вызывает фильтрацию.

**Модуль подсчета записей (def** rowCount\_in\_statusBar(self))

По умолчанию в модель загружается только 256 записей. Чтобы узнать количество записей или узнать все значения, необходимо загрузить все данные.

Модуль осуществляет загрузку всех данных в модель и выводит их количество в statusBar.

**Презентацию**

(если вы проводите защиту проекта)

**Видео**

(при необходимости, например, для проекта PyGame)