**МИНИСТЕРСТВО ОБРАЗОВАНИЯ МОСКОВСКОЙ ОБЛАСТИ**

**Государственное бюджетное профессиональное образовательное учреждение Московской области «Люберецкий техникум имени Героя Советского Союза, летчика-космонавта Ю.А.Гагарина»**

**КУРСОВАЯ РАБОТА**

ПМ.01 МДК 01.04 Эксплуатация автоматизированных (информационных) систем в защищенном исполнении.

По теме: Создать базу данных автоматизации учёта и прогноза запасов нефтепродуктов, реализуемых нефтебазой.

Выполнил студент \_\_\_\_\_\_\_\_\_\_\_ Ефремов Егор Михайлович

(подпись) (ФИО полностью)

Руководитель Тарджиманян Лия Николаевна

(подпись) (ФИО полностью)

Оценка \_\_\_\_\_\_\_\_\_\_\_\_\_\_

Дзержинский 2023

Содержание

Введение

ГЛАВА 1. Теоретические сведения о БД.

1.1 Реляционные базы данных

1.2 Описание базы данных SQLite

ГЛАВА 2. Создание Базы данных

2.1 Физическая структура

2.2 Работа с данными

2.3 Описание информационных потребностей пользователей и выбор способов их реализации.

2.4 Работа с формами базы данных

2.5 Разработка интерфейса

2.6 Создание формы

Заключение

Список источников

**ВВЕДЕНИЕ**

В данной работе будет рассматриваться создание Базы данных и графического интерфейса для работы с ней.

В современном мире базы данных используются повсеместно. Это обусловлено необходимостью работать с большими объемами информации практически во всех областях деятельности. Базы данных позволяют структурировать и систематизировать информацию, что облегчает и ускоряет работу с ней. Соответственно, перед многими работниками ставятся задачи по использованию баз данных. Понимание основ подключения к базам данных и работе с ними является актуальным вопросом. Далее в работе будут рассмотрены основы работы с базами данных на примере баз данных SQLite.

Прогресс не стоит на месте, информационные и компьютерные технологии развиваются, расширяются. Современного пользователя уже не устроит программа, работающая в консоли и управляемая текстовыми командами. Теперь программа должна быть не только эффективной, но и максимально удобной в использовании для любого человека, вне зависимости от его уровня знаний в области информационных технологий и программирования. Задача разработчика – обеспечить для рядового пользователя возможность управлять базой данных, не вникая в весь процесс взаимодействия с ней. Для выполнения этой задачи существует графический интерфейс пользователя –Graphical User Interface или, сокращенно, GUI. При создании GUI-приложения необходимо уметь не только создать удобную графическую оболочку для программы, но и связать эту оболочку с действиями, которые будет выполнять программа. В данной работе будут рассмотрены несколько базовых элементов GUI и их привязка к конкретным действиям.

**Цель:** создание базы данных и графического интерфейса на языке Python, которая будет работать с базой данных, содержащей информацию о автоматизации учёта и прогноза запасов нефтепродуктов, реализуемых нефтебазой. . В рамках поставленной цели будут рассмотрены и реализованы следующие задачи:

* создание базы данных;
* создание графического интерфейса;
* разработка визуального интерфейса;

Объектом исследования данной работы является создание базы данных с визуальным интерфейсом.

**ГЛАВА 1 ТЕОРЕТИЧЕСКИЕ СВЕДЕНИЯ ПО БАЗЫ ДАННЫХ**

**1.1 Реляционные базы данных**

Базы данных (далее БД) создаются для систематизирования и структурирования больших объемов информации, чтобы потом извлекать необходимые данные быстро и в любых сочетаниях. Реляционная база данных – это связанная информация, хранящаяся в двумерных таблицах. Ячейки таблицы являются пересечением строк и столбцов. В БД каждый столбец имеет свое ключевое имя, по которому и происходит обращение к нужному столбцу или элементу в столбце.

Системы управления базами данных (СУБД) – это программное обеспечение для создания, поддержки БД и доступа к ней. В данной работе будет использоваться СУБД SQLite3, так как язык программирования Python имеет встроенный модуль для работы с SQLite3 и она является наиболее удобной для знакомства с программированием баз данных. SQLite3 поддерживает следующие типы данных: целые числа (integer), вещественные числа (real), строковый тип (text), бинарные данные (blob).

Доступ к БД осуществляется с помощью объекта-соединения. Объект-соединение получают путем вызова функции connect( ). Если указанной БД для него не существует, то она будет создана. Объект-соединение имеет следующие методы:

* close( );
* commit( );
* rollback( );
* cursor( ).

Метод close( ) закрывает соединение с текущей БД .

Метод commit( ) завершает транзакцию и сохраняет изменения.

Транзакция – это одна или несколько операций, изменяющих БД .

Метод rollback( ) откатывает начатую транзакцию и восстанавливает исходное состояние БД.

Метод cursor( ) возвращает объект-курсор, использующий данное соединение. Курсор хранит информацию о текущей таблице и записи, может быть связан с выполнением только одного SQL-запроса в текущий момент времени, служит для работы с результатами запроса .

Для работы с БД в программе будет использован атрибут объекта курсора execute( ), который преобразовывает SQL-команду в строке в исполняемый код.

Основные SQL-команды:

* создание таблицы в БД – CREATE TABLE <имя таблицы> (<имя поля 1 тип поля>, <имя поля 2 тип поля>, …);
* добавление записи в таблицу БД – INSERT INTO <имя таблицы> VALUES (‘<значение1>’, ‘<значение2>’, …);
* выборка из таблицы БД по условию – SELECT (список полей, включенных в выборку) FROM <имя таблицы> WHERE <имя поля > = <значение>;
* выборка из таблицы БД с сортировкой – SELECT (список полей, включенных в выборку) FROM <имя таблицы> ORDER BY <имя поля, по которому производится сортировка>;
* удаление записей из таблицы БД по условию – DELETE FROM <имя таблицы> WHERE <имя поля> = <значение>;
* изменение записей в таблице БД по условию – UPDATE <имя таблицы> SET <имя изменяемого поля> = <новое значение> WHERE <имя поля> = <значение>.

Результаты выборок передаются в виде кортежей.

**1.2 Описание базы данных SQLite**

SQLite — компактная встраиваемая реляционная база данных. Исходный код библиотеки передан в общественное достояние. Является чисто реляционной базой данных.

Слово «встраиваемый» означает, что SQLite не использует парадигму клиент-сервер. То есть движок SQLite не является отдельно работающим процессом, с которым взаимодействует программа, а предоставляет библиотеку, с которой программа компонуется и движок становится составной частью программы. Таким образом, в качестве протокола обмена используются вызовы функций (API) библиотеки SQLite. Такой подход уменьшает накладные расходы, время отклика и упрощает программу. SQLite хранит всю базу данных (включая определения, таблицы, индексы и данные) в единственном стандартном файле на том компьютере, на котором исполняется программа. Простота реализации достигается за счёт того, что перед началом исполнения транзакции записи весь файл, хранящий базу данных, блокируется; ACID функции достигаются в том числе за счет создания файла журнала.

Несколько процессов или потоков могут одновременно без каких-либо проблем читать данные из одной базы. Запись в базу можно осуществить только в том случае, если никаких других запросов в данный момент не обслуживается; в противном случае попытка записи оканчивается неудачей, и в программу возвращается код ошибки. Другим вариантом развития событий является автоматическое повторение попыток записи в течение заданного интервала времени. Можно, также, ввести таймаут операций. Тогда подключение, столкнувшись с занятостью БД, будет ждать N секунду прежде, чем отвалиться с ошибкой SQLITE\_BUSY.

Также с версии 3.7.0 присутствует режим WAL , с помощью которого можно использовать одну и ту же базу несколькими приложениями, как на чтение, так и на запись.

В комплекте поставки идет также функциональная клиентская часть в виде исполняемого файла sqlite3, с помощью которого демонстрируется реализация функций основной библиотеки. Клиентская часть работает из командной строки, позволяет обращаться к файлу БД на основе типовых функций ОС.

Благодаря архитектуре движка возможно использовать SQLite как на встраиваемых системах, так и на выделенных машинах с гигабитными массивами данных.

Формат файла базы данных является кросс-платформенным, что позволяет без проблем использовать одну и ту же базу на нескольких операционных системах. Также присутствует возможность хранения базы в памяти, без её записи на диск. Этот вариант используется по умолчанию для консольной утилиты sqlite3, если не указано имя файла.

## *Типизация -* SQLite использует динамическое типизирование данных. Это значит, что тип столбца не определяет тип хранимого значения в этом поле записи. Т.е. в любой столбец можно ввести любое значение.

Тип столбца определяет как сравнивать значения (нужно же их привести к единому типу при сравнении, скажем, внутри индекса). Но не обязывает заносить значения именно такого типа в столбец.

Допустим, мы объявили столбец как «A INTEGER». SQLite позволяет внести в этот столбец значения любого типа (999, «abc», «123», 678.525). Если вставляемое значение — не целое, то SQLite пытается привести его к целому. Т.е. строка «123» превратится в целое 123, а остальные значения запишутся «как есть».

Возможные типы полей: NULL, INTEGER, REAL, TEXT, BLOB.

## *Надежность -* Ситуация с покрытиями тестами исходного кода SQLite в некотором роде является легендой.

Это связно с тем, что тестами описано практически все возможные (и невозможные) ситуации. Кода, описывающего тесты, намного больше, чем кода реализующего SQLite. Естественно, что покрытие кода тестами 100%.

Сам же подход к тестированию использующийся разработчиками SQLite достоин подражания. И может быть взят как идеал, к которому следуют стремиться.

**ГЛАВА 2. Создать базу данных автоматизации учёта и прогноза запасов нефтепродуктов, реализуемых нефтебазой.  
2.1 Физическая структура**

Таблица “Сводная” имеет следующие атрибуты:

* id
* id\_AZS
* date\_usl
* time\_usl
* id\_product

Таблица “АЗС” (Clients) имеет следующие атрибуты:

* Id
* IdAzs
* addres
* Phone

Таблица “Продукт” (Uslugi) имеет следующие атрибуты:

* id
* Azs
* Product
* Purchases
* Rests
* Sales
* Supplier

**2.2 Разработка графического интерфейса.**

Для начала в QT 5 дизайнере необходимо создать внешний вид форм, перенести на окно необходимые кнопки, поля для ввода данных, текстовые значения. После создания внешнего вида формы нам необходимо добавить функции на кнопки и поля ввода, чтобы наш интерфейс исправно функционировал.

***Назначение команд для функционирования форм***

Импортирование необходимых библиотек

import sys

import sqlite3

from PyQt5 import QtWidgets

from PyQt5.QtWidgets import QDialog, QApplication, QTableWidgetItem

from PyQt5.uic import loadUi

*Подключение к бд*

db = sqlite3.connect("bd.db")

sql = db.cursor()

*Вход в учётную запись*

class Login(QDialog):

def \_\_init\_\_(self):

super(Login,self).\_\_init\_\_()

loadUi("login.ui",self)

self.password.setEchoMode(QtWidgets.QLineEdit.Password)

self.loginbutton.clicked.connect(self.loginfunction)

self.createaccbutton.clicked.connect(self.gotocreate)

def loginfunction(self):

*Запрос у пользователя данных для входа*

email=self.email.text()

password=self.password.text()

Поиск данных, которые ввел пользователь, в бд

sql.execute(f"SELECT \* FROM users WHERE username = '{self.email.text()}' AND password = '{self.password.text()}';")

db.commit()

*Цикл на авторизацию*

if sql.fetchone() == None:

print("Нет такой записи")

else:

print('Welcome')

loginbutton=Client()

widget.addWidget(loginbutton)

widget.setCurrentIndex(widget.currentIndex()+1)

*Кнопка, которая отправляет пользователя на страницу регистрации*

def gotocreate(self):

createaccbutton=CreateAcc()

widget.addWidget(createaccbutton)

widget.setCurrentIndex(widget.currentIndex()+1)

*Регистрация*

class CreateAcc(QDialog):

def \_\_init\_\_(self):

super(CreateAcc,self).\_\_init\_\_()

loadUi("createacc.ui",self)

self.signupbutton.clicked.connect(self.createaccfunction)

self.password.setEchoMode(QtWidgets.QLineEdit.Password)

self.confirmpass.setEchoMode(QtWidgets.QLineEdit.Password)

self.conn = None

def createaccfunction(self):

*Запрос на ввод данных от пользователя*

email = self.email.text()

*Поиск данных, которые ввел пользователь*

password=self.password.text()

if self.password.text()==self.confirmpass.text():

*Запрос на поиск информации, которую ввел пользователь*

sql.execute(f"SELECT username, password FROM users WHERE username = '{email}' AND password = '{password}'")

*Цикл на регистрацию*

if sql.fetchone() is None:

*Запрос на добавление новых записей в таблицу для регистрации пользователя*

sql.execute(f"INSERT INTO users VALUES (?,?)", (email, password))

db.commit()

print('You have registered')

signupbutton=Login()

widget.addWidget(signupbutton)

widget.setCurrentIndex(widget.currentIndex()+1)

else:

print('Такая запись уже существует')

for i in sql.execute('SELECT \* FROM users'):

print(i)

**2.3 Работа с формами базы данных**

*Изменение данных в таблице:*

Для того, чтобы изменить данные в таблице необходимо навести курсор на ячейку таблицы, в которой Вы хотите произвести изменения и несколько раз нажать на нее, после чего изменить данные.

*Удаление записей:*

Если данные были введены неверно, то их можно удалить из базы данных. Для того, чтобы удалить данные из базы данных необходимо в нижнем окне, где отображается таблица, навести на номер записи, которую Вы хотите удалить и нажать на кнопку “Удалить”. которая располагается в одной колонке с кнопкой “Открыть”.

*Поиск по определенным критериям:*

Также через формы можно выводить данные, которые нужно отсортировать по тому или иному критерию. Для этого в разделе поиска нажимаем на выпадающее окно, где представлены названия столбцов, и выбираем назание столбца, по значением которого нам нужно вывести данные. Ниже, в окошке для ввода данных нужно указать значения, по которым мы будем сортировать и нажимаем кнопку “Найти”. В окошко, через которое отображается таблица, будут выведены необходимые отсортированные данные.

*Переключение между формами:*

Чтобы переключаться между формами предусмотрены кнопки, которые располагаются под блоком “Поиск”. Нажимая на одну из кнопок вы будете переходить на другую форму. Все формы работаю аналогично форме “Клиенты”.

**2.4 Описание информационных потребностей пользователей и выбор способов их реализации.**

У пользователя могут возникнуть следующие потребности при пользовании интерфейсом:

* Регистрация на форме
* Авторизация на форме
* Изменение данных в таблице
* Удаление данных из базы данных
* Поиск отсортированной информации и ее вывод

*Авторизация:*

Если Вы уже зарегистрированы для работы с формами(интерфейсом), то Вам необходимо просто внести данные в соответствующие поля без ошибок и Вас перенесет на страницу “Клиенты”...

Но если Вы не зарегистрированы, то перед началом работы с формами(интерфейсом) нужно зарегистрироваться. Это можно сделать нажав на кнопку “регистрация” и ввести свой адрес электронной почты и пароль в соответствующие ячейки. Во вторую строку “пароль” нужно ввести пароль, который Вы указали выше. После всех выполненных действий происходит выполнение авторизации.

**2.5 Работа с данными**

*Организация ввода данных в БД.*

Первичный ввод записей в базу данных будет производится при помощи импортирования данных из Excel. Далее ввод данных будет производится непосредственно с формы.

*Организация корректировки БД*.

Корректироваться данных производится с формы, или непосредственно в Sqlite.

**2.6 Создание формы**

class Client(QDialog):

def \_\_init\_\_(self):

super(Client, self).\_\_init\_\_()

loadUi("client.ui",self)

self.rbMale.setChecked(True)

self.pbInsert.clicked.connect(self.insert\_staff)

self.pbOpen.clicked.connect(self.open\_file)

self.pbDelete.clicked.connect(self.delete\_staff)

self.pbFind.clicked.connect(self.find\_for\_val)

self.pushButton\_3.clicked.connect(self.gotouslugu)

self.pushButton\_2.clicked.connect(self.gotosvod)

self.conn = None

*Кнопк открытия БД*

def open\_file(self):

try:

self.conn = sqlite3.connect('school(1).db')

cur = self.conn.cursor()

*Запрос на вывод таблицы*

data = cur.execute("select \* from client")

col\_name = [i[0] for i in data.description]

data\_rows = data.fetchall()

except Exception as e:

print(f"Проблемы с подключением к БД. {e}")

return e

self.twStaffs.setColumnCount(len(col\_name))

self.twStaffs.setHorizontalHeaderLabels(col\_name)

self.twStaffs.setRowCount(0)

self.cbColNames.addItems(col\_name)

for i, row in enumerate(data\_rows):

self.twStaffs.setRowCount(self.twStaffs.rowCount() + 1)

for j, elem in enumerate(row):

self.twStaffs.setItem(i, j, QTableWidgetItem(str(elem)))

self.twStaffs.resizeColumnsToContents()

self.avg\_age()

*Изменение данных в таблице и ее вывод*

def update\_twStaffs(self, query="select \* from client"):

try:

cur = self.conn.cursor()

data = cur.execute(query).fetchall()

except Exception as e:

print(f"Проблемы с подключением к БД. {e}")

return e

self.twStaffs.setRowCount(0)

for i, row in enumerate(data):

self.twStaffs.setRowCount(self.twStaffs.rowCount() + 1)

for j, elem in enumerate(row):

self.twStaffs.setItem(i, j, QTableWidgetItem(str(elem)))

self.twStaffs.resizeColumnsToContents()

*Запрос на удаление данных из таблицы*

cur.execute(f"delete from client where id = {num}")

self.conn.commit()

cur.close()

except Exception as e:

print(f"Исключение: {e}")

return e

self.update\_twStaffs()

*Поиск информации с сортировкой*

def find\_for\_val(self):

val = self.leFind.text()

col = self.cbColNames.itemText(self.cbColNames.currentIndex())

Запрос на сортировку

self.update\_twStaffs(f"select \* from client where {col} like '{val}%'")

def closeEvent(self, event):

if self.conn is not None:

self.conn.close()

event.accept()

*Кнопк, которая переходит на форму “Сводная”*

def gotosvod(self):

pushButton\_2=Swod()

widget.addWidget(pushButton\_2)

widget.setCurrentIndex(widget.currentIndex()+1)

*Кнопк, которая переходит на форму “Услуги”*

def gotouslugu(self):

pushButton\_3=Uslugu()

widget.addWidget(pushButton\_3)

widget.setCurrentIndex(widget.currentIndex()+1)

**ЗАКЛЮЧЕНИЕ**

**Целью** данной курсовой работы являлось создание базы данных и графического интерфейса на языке Python, содержащей информацию об автоматизации учёта и прогноза запасов нефтепродуктов, реализуемых нефтебазой. В рамках поставленной цели были рассмотрены и реализованы следующие задачи:

* была создана база данных SQLite;
* был разработан графический интерфейса PyQt5;
* была реализована разработка визуального интерфейса;

Что позволило создать полностью функционирующую базу данных на платформе SQLite с разработанным графическим интерфейсом.

**СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ**

1. **Python GUI: создаём простое приложение с PyQt и Qt Designer** <https://tproger.ru/translations/python-gui-pyqt/> (дата обращения: 28.12.2022г )

# **Руководство по SQLite в Python** <https://pythonru.com/osnovy/sqlite-v-python>(дата обращения: 17.12.2022г)

# **PyQt5: первые программы** <https://pythonworld.ru/gui/pyqt5-firstprograms.html>(дата обращения: 20.12.2022г)