СОДЕРЖАНИЕ

[ВВЕДЕНИЕ 5](#_Toc27949227)

[ОСНОВНАЯ ЧАСТЬ 8](#_Toc27949228)

[**Структуры баз данных** 8](#_Toc27949229)

[**Описание созданной СУБД** 10](#_Toc27949230)

[ВЫВОД 28](#_Toc27949231)

[СПИСОК ИСПОЛЬЗОВАННОЙ ЛИТЕРАТУРЫ 29](#_Toc27949232)

# **ВВЕДЕНИЕ**

В настоящее время, жизнь общества уже не представима без компьютеров и вычислительных машин, все сферы жизни построены на работе с ними. Их использования позволяет упростить и/или улучшить практически любую часть работы, любую часть человеческой нужды. Так, для использования мощностей компьютерной технике возникли языки программирования. Python – это высокоуровневый язык программирования, синтаксис ядра которого минималистичен, но и его стандартная библиотека включает огромный объём полезных функций. Python поддерживает структурное, объектно-ориентированное, функциональное, императивное и аспектно-ориентированное программирование. Основные архитектурные черты — динамическая типизация, автоматическое управление памятью, полная интроспекция, механизм обработки исключений, поддержка многопоточных вычислений, высокоуровневые структуры данных. Поддерживается разбиение программ на модули, которые, в свою очередь, могут объединяться в пакеты. Стоит отметить, что Python – это активно развивающийся язык, обновления на него выходят примерно раз в два года. Разработка языка Python была начата в конце 1980-х годов сотрудником голландского института CWI Гвидо ван Россумом. Для распределённой ОС Amoeba требовался расширяемый скриптовый язык, и Гвидо начал писать Python на досуге, позаимствовав некоторые наработки для языка ABC (Гвидо участвовал в разработке этого языка, ориентированного на обучение программированию). С самого начала Python проектировался как объектно-ориентированный язык. Название языка произошло вовсе не от вида пресмыкающихся. Автор назвал язык в честь популярного британского комедийного телешоу 1970-х «Летающий цирк Монти Пайтона». Впрочем, всё равно название языка чаще связывают именно со змеёй, нежели с передачей — пиктограммы файлов в KDE или в Microsoft Windows и даже эмблема на сайте python.org (до выхода версии 2.5) изображают змеиные головы. Важная цель разработчиков Python — создавать его забавным для использования. Это отражено в его названии, которое пришло из Монти Пайтона. Также это отражено в иногда игривом подходе к обучающим программам и справочным материалам, таким как примеры использования, которые используют понятия ветчины (spam) и яиц вместо стандартных foo и bar.

Наличие дружелюбного, отзывчивого сообщества пользователей считается наряду с дизайнерской интуицией Гвидо одним из факторов успеха Python. Развитие языка происходит согласно чётко регламентированному процессу создания, обсуждения, отбора и реализации документов PEP (англ. Python Enhancement Proposal) — предложений по развитию Python. 3 декабря 2008 года, после длительного тестирования, вышла первая версия Python 3000 (или Python 3.0, также используется сокращение Py3k). В Python 3000 устранены многие недостатки архитектуры с максимально возможным (но не полным) сохранением совместимости со старыми версиями Python. На сегодня поддерживаются обе ветви развития (Python 3.x и 2.x).

База данных – это представленная в объективной форме совокупность самостоятельных материалов (статей, расчётов, нормативных актов, судебных решений и иных подобных материалов), систематизированных таким образом, чтобы эти материалы могли быть найдены и обработаны с помощью электронной вычислительной машины (ЭВМ).

Многие специалисты указывают на распространённую ошибку, состоящую в некорректном использовании термина «база данных» вместо термина «система управления базами данных», и указывают на необходимость различения этих понятий.

Определения из международных стандартов и национальных стандартов, разработанных на основе международных:

База данных — совокупность данных, хранимых в соответствии со схемой данных, манипулирование которыми выполняют в соответствии с правилами средств моделирования данных.

База данных — совокупность данных, организованных в соответствии с концептуальной структурой, описывающей характеристики этих данных и взаимоотношения между ними, которая поддерживает одну или более областей применения.

История возникновения и развития технологий баз данных может рассматриваться как в широком, так и в узком аспекте.

В широком смысле понятие истории баз данных обобщается до истории любых средств, с помощью которых человечество хранило и обрабатывало данные. В таком контексте упоминаются, например, средства учёта царской казны и налогов в древнем Шумере (4000 г. до н. э.), узелковая письменность инков — кипу, клинописи, содержащие документы Ассирийского царства и т. п. Следует помнить, что недостатком этого подхода является размывание понятия «база данных» и фактическое его слияние с понятиями «архив» и даже «письменность». История баз данных в узком смысле рассматривает базы данных в традиционном (современном) понимании. Эта история начинается с 1955 года, когда появилось программируемое оборудование обработки записей. Программное обеспечение этого времени поддерживало модель обработки записей на основе файлов. Для хранения данных использовались перфокарты. Оперативные сетевые базы данных появились в середине 1960-х. Операции над оперативными базами данных обрабатывались в интерактивном режиме с помощью терминалов. Простые индексно-последовательные организации записей быстро развились к более мощной модели записей, ориентированной на наборы. За руководство работой Data Base Task Group (DBTG), разработавшей стандартный язык описания данных и манипулирования данными, Чарльз Бахман получил Тьюринговскую премию. В это же время в сообществе баз данных Кобол была проработана концепция схем баз данных и концепция независимости данных. Следующий важный этап связан с появлением в начале 1970-х реляционной модели данных, благодаря работам Эдгара Кодда. Работы Кодда открыли путь к тесной связи прикладной технологии баз данных с математикой и логикой. За свой вклад в теорию и практику Эдгар Ф. Кодд также получил премию Тьюринга.

Сам термин база данных (англ. database) появился в начале 1960-х годов, и был введён в употребление на симпозиумах, организованных компанией SDC в 1964 и 1965 годах, хотя понимался сначала в довольно узком смысле, в контексте систем искусственного интеллекта. В широкое употребление в современном понимании термин вошёл лишь в 1970-е годы.

Также пару слов стоит сказать о системах управления базами данных и что это такое:

Систе́ма управле́ния ба́зами да́нных, сокр. СУБД (англ. Database Management System, сокр. DBMS) — совокупность программных и лингвистических средств общего или специального назначения, обеспечивающих управление созданием и использованием баз данных.

СУБД — ком­плекс про­грамм, по­зво­ляю­щих соз­дать ба­зу дан­ных (БД) и ма­ни­пу­ли­ро­вать дан­ны­ми (встав­лять, об­нов­лять, уда­лять и вы­би­рать). Система обес­пе­чи­ва­ет безо­пас­ность, на­дёж­ность хра­не­ния и це­ло­ст­ность дан­ных, а так­же пре­дос­тав­ля­ет сред­ст­ва для ад­ми­ни­ст­ри­ро­ва­ния БД.

# **ОСНОВНАЯ ЧАСТЬ**

## **Структуры баз данных**

При работе с базами данных, мы стремимся упорядочить информацию по различным признакам для того, чтобы потом извлекать из нее необходимые нам данные в любом сочетании. Сделать это возможно, только если данные структурированы. Структурирование — это набор соглашений о способах представления данных. Понятно, что структурировать информацию можно по-разному. В зависимости от структуры различают иерархическую, сетевую, реляционную, объектно-ориентированную и гибридную модели баз данных. Самой популярной на сегодняшний день является реляционная структура. Для сравнения опишем реляционную структуру и иерархическую.

Иерархическая модель данных — это модель данных, где используется представление базы данных в виде древовидной (иерархической) структуры, состоящей из объектов (данных) различных уровней. Между объектами существуют связи, каждый объект может включать в себя несколько объектов более низкого уровня. Такие объекты находятся в отношении предка (объект более близкий к корню) к потомку (объект более низкого уровня), при этом возможна ситуация, когда объект-предок имеет несколько потомков, тогда как у объекта-потомка обязателен только один предок. Объекты, имеющие общего предка, называются близнецами (в программировании применительно к структуре данных дерево устоялось название братья). Базы данных с иерархической моделью одни из самых старых, и стали первыми системами управления базами данных для мейнфреймов.

Основными информационными единицами в иерархической модели данных являются сегмент и поле. Поле данных определяется как наименьшая неделимая единица данных, доступная пользователю. Для сегмента определяются тип сегмента и экземпляр сегмента. Экземпляр сегмента образуется из конкретных значений полей данных. Тип сегмента — это поименованная совокупность входящих в него типов полей данных.

Как и сетевая, иерархическая модель данных базируется на графовой форме построения данных, и на концептуальном уровне она является просто частным случаем сетевой модели данных. В иерархической модели данных вершине графа соответствует тип сегмента или просто сегмент, а дугам — типы связей предок — потомок. В иерархических структуpax сегмент — потомок должен иметь в точности одного предка. Иерархическая модель представляет собой связный неориентированный граф древовидной структуры, объединяющий сегменты. Иерархическая БД состоит из упорядоченного набора деревьев.

Теперь скажем несколько слов о реляционной базе данных.

Реляционная модель данных (РМД) — логическая модель данных, прикладная теория построения баз данных, которая является приложением к задачам обработки данных таких разделов математики, как теория множеств и логика первого порядка.

Реляционная модель данных включает следующие компоненты:

- Структурный аспект (составляющая) — данные в базе данных представляют собой набор отношений.

- Аспект (составляющая) целостности — отношения отвечают определённым условиям целостности. РМД поддерживает декларативные ограничения целостности уровня домена (типа данных), уровня отношения и уровня базы данных.

- Аспект (составляющая) обработки (манипулирования) — РМД поддерживает операторы манипулирования отношениями (реляционная алгебра, реляционное исчисление).

Кроме того, в состав реляционной модели данных включают теорию нормализации. Термин «реляционный» означает, что теория основана на математическом понятии отношение (relation). В качестве неформального синонима термину «отношение» часто встречается слово таблица. Необходимо помнить, что «таблица» есть понятие нестрогое и неформальное и часто означает не «отношение» как абстрактное понятие, а визуальное представление отношения на бумаге или экране. Некорректное и нестрогое использование термина «таблица» вместо термина «отношение» нередко приводит к недопониманию. Наиболее частая ошибка состоит в рассуждениях о том, что РМД имеет дело с «плоскими», или «двумерными» таблицами, тогда как таковыми могут быть только визуальные представления таблиц. Отношения же являются абстракциями и не могут быть ни «плоскими», ни «неплоскими».

Для лучшего понимания РМД следует отметить три важных обстоятельства:

- модель является логической, то есть отношения являются логическими (абстрактными), а не физическими (хранимыми) структурами;

- для реляционных баз данных верен информационный принцип: всё информационное наполнение базы данных представлено одним и только одним способом, а именно — явным заданием значений атрибутов в кортежах отношений; в частности, нет никаких указателей (адресов), связывающих одно значение с другим;

- наличие реляционной алгебры позволяет реализовать декларативное программирование и декларативное описание ограничений целостности, в дополнение к навигационному (процедурному) программированию и процедурной проверке условий.

## **Описание созданной СУБД**

В дальнейшей работе будет использована библиотека PyQt5 для проектирования приложения (дизайн), и написано приложение на языке Python для работы с БД. Также будет использована библиотека psycopg2, позволяющая “общаться” с БД на PostgreSQL напрямую через код программы. Сначала приведем листинг программы, после которого опишем поподробнее некоторые моменты.

После работы с дизайнером PyQt5 на выходе мы имеем файлы с расширением .ui, воспользовавшись командной строкой и pyuic5 конвертируем данные файлы в “питоновские” с расширением .py, в которых уже можем работать с кодом. Были созданы несколько файлов: DataBaseGUI – в котором содержится главное окно программы с меню и прочими возможностями.

Connection – окно подключения к базе данных.

Create – окно с интерфейсом для создания таблиц.

Cortege – окно для добавления строки в таблицу.

Attribute – окно для добавления столбца в таблицу.

Change – окно с возможностью изменения строк в таблице при выполнении определенного условия.

Select – окно с выбором условий вывода столбцов из таблиц.

Select\_Table – окно с непосредственным выводом данных из предыдущего пункта.  
  
Файл main.py

|  |  |
| --- | --- |
| import psycopg2  from psycopg2 import sql  import sys  import csv  import random  from PyQt5 import QtWidgets, QtCore  from PyQt5.QtWidgets import QApplication, QWidget, QFileDialog, QInputDialog, QTableWidgetItem, QMessageBox  from DataBaseGUI import Ui\_MainWindow  import Connection  import Create  import Cortege  import Attribute  import Change  import Select  import Select\_Table  class DB\_GUI(QtWidgets.QMainWindow):  def \_\_init\_\_(self):  super(DB\_GUI, self).\_\_init\_\_()  self.ui = Ui\_MainWindow()  self.ui.setupUi(self)  self.sql\_command = {}  self.table = []  self.table\_header = []  self.table\_name = ""  self.flag = True    self.sql\_command["create"] = "CREATE TABLE {} ({});"  self.sql\_command["select"] = "SELECT {} FROM {};"  self.sql\_command["selwhe"] = "SELECT {} FROM {} WHERE {};"  self.sql\_command["insert"] = "INSERT INTO {} VALUES ({});"  self.sql\_command["addcol"] = "ALTER TABLE {} ADD COLUMN {} {};"  self.sql\_command["dropcol"] = "ALTER TABLE {} DROP {};"  self.sql\_command["delete"] = "DELETE FROM {} WHERE {};"  self.sql\_command["drop"] = "DROP TABLE IF EXISTS {};"  self.sql\_command["sample"] = "SELECT {} FROM {} WHERE {};"  self.sql\_command["right"] = "select a.attname FROM pg\_catalog.pg\_attribute a inner join pg\_catalog.pg\_class c on a.attrelid = c.oid where c.relname = '{}' and a.attnum > 0 and a.attisdropped=false and pg\_catalog.pg\_table\_is\_visible(c.oid) order by a.attnum;"  self.sql\_command["change"] = "UPDATE {} SET {} = {} WHERE {};"  self.sql\_command["copy"] = "COPY {} FROM {!r} DELIMITER ';' ENCODING 'WIN1251' CSV HEADER";  self.ui.action\_3.triggered.connect(self.save\_as)  self.ui.action\_4.triggered.connect(self.close\_app)  self.ui.action\_5.triggered.connect(self.showDlgCreateDB)  self.ui.action\_10.triggered.connect(self.showDlgCortegeAdd)  self.ui.action\_11.triggered.connect(self.delete\_cortege\_db)  self.ui.action\_6.triggered.connect(self.delete\_db)  self.ui.action\_9.triggered.connect(self.showDlgConnectionDB)  self.ui.actionAttribute.triggered.connect(self.showDlgAttributeAdd)  self.ui.actionAttribute\_2.triggered.connect(self.delete\_attribute\_db)  self.ui.pushButton.clicked.connect(self.change\_cortege)  self.ui.pushButton2.clicked.connect(self.selects)    def showDlgConnectionDB(self):  self.dlg = QtWidgets.QDialog()  self.dlg\_ui = Connection.Ui\_Dialog()  self.dlg\_ui.setupUi(self.dlg)  self.dlg.show()  self.dlg\_ui.buttonBox.accepted.connect(self.connect\_db)  self.dlg\_ui.buttonBox.rejected.connect(self.dlg.close)  def change\_cortege(self):  self.dlgc = QtWidgets.QDialog()  self.dlgc\_ui = Change.Ui\_Dialog()  self.dlgc\_ui.setupUi(self.dlgc)  self.dlgc.show()    self.dlgc\_ui.buttonBox.accepted.connect(self.changeok)  self.dlgc\_ui.buttonBox.rejected.connect(self.dlgc.close)  def changeok(self):  try:  changes\_attr=self.dlgc\_ui.lineEdit.text()  usl=self.dlgc\_ui.lineEdit\_2.text()  value=self.dlgc\_ui.lineEdit\_3.text()  if changes\_attr:  insert = sql.SQL(self.sql\_command["change"].format(self.table\_name, changes\_attr,value,usl))  self.cursor.execute(insert)  self.get\_table()  except:  QMessageBox.critical(self, "Error", "Unknown error")  def selects(self):  self.dlgc = QtWidgets.QDialog()  self.dlgc\_ui = Select.Ui\_Dialog()  self.dlgc\_ui.setupUi(self.dlgc)  self.dlgc.show()    self.dlgc\_ui.buttonBox.accepted.connect(self.selectw)  self.dlgc\_ui.buttonBox.rejected.connect(self.dlgc.close)  attr='req'  def selectw(self):  try:  attr=self.dlgc\_ui.lineEdit\_2.text()  usl=self.dlgc\_ui.lineEdit.text()  if attr:  self.sel\_table()  except:  QMessageBox.critical(self, "Error", "Unknown error")  def sel\_table(self):  self.dlgcs = QtWidgets.QDialog()  self.dlgcs\_ui = Select\_Table.Ui\_Dialog()  self.dlgcs\_ui.setupUi(self.dlgcs)  s=self.dlgc\_ui.lineEdit\_2.text()  u=self.dlgc\_ui.lineEdit.text()  if self.table\_name:  self.cursor.execute(self.sql\_command["selwhe"].format(s, self.table\_name,u))  self.table = []  i=0  a=s.split(',')  print(a)  k=0  t=0  l=len(self.table\_header)  while i < l:  while t < len(a):  if self.table\_header[i]==a[t]:  k=1  t=t+1  t=0  if k==1:  for j in self.cursor:  self.table.append(j)  else:  self.table\_header.pop(i)  l=len(self.table\_header)  continue;  i=i+1  k=0    self.dlgcs\_ui.tableWidget.setRowCount(len(self.table))  self.dlgcs\_ui.tableWidget.setColumnCount(len(self.table\_header))  self.dlgcs\_ui.tableWidget.setHorizontalHeaderLabels(self.table\_header)  row = 0  for tup in self.table:  col = 0  for item in tup:  cellinfo = QTableWidgetItem(str(item))  cellinfo.setFlags(QtCore.Qt.ItemIsSelectable | QtCore.Qt.ItemIsEnabled)  self.dlgcs\_ui.tableWidget.setItem(row, col, cellinfo)  col += 1  row += 1  self.dlgcs.show()  self.attribute\_list()  self.dlgcs\_ui.buttonBox.accepted.connect(self.selectw)  self.dlgcs\_ui.buttonBox.rejected.connect(self.dlgcs.close)    def connect\_db(self):  self.dlg.close()  try:  self.conn = psycopg2.connect(dbname=self.dlg\_ui.lineEdit.text(),  user=self.dlg\_ui.lineEdit\_2.text(),  password=self.dlg\_ui.lineEdit\_3.text(),  host=self.dlg\_ui.lineEdit\_4.text())  self.conn.autocommit = True  self.cursor = self.conn.cursor()  self.cursor.execute("select table\_name from information\_schema.tables where table\_schema \  not in ('information\_schema','pg\_catalog');")  self.init\_comboBox()  self.ui.label\_3.setText(self.dlg\_ui.lineEdit.text())  except:  QMessageBox.critical(self, "Error", "Unknown error")  def update\_table\_name(self, name):  self.table\_name = name  self.get\_table()  def attribute\_list(self):  self.cursor.execute(self.sql\_command["right"].format(self.table\_name))  self.table\_header = []  for i in self.cursor:  print(self.table\_header)  self.table\_header.append(i[0])  def update\_table(self):  self.attribute\_list()  self.ui.tableWidget.setRowCount(len(self.table))  self.ui.tableWidget.setColumnCount(len(self.table\_header))  self.ui.tableWidget.setHorizontalHeaderLabels(self.table\_header)  row = 0  for tup in self.table:  col = 0  for item in tup:  cellinfo = QTableWidgetItem(str(item))  cellinfo.setFlags(QtCore.Qt.ItemIsSelectable | QtCore.Qt.ItemIsEnabled)  self.ui.tableWidget.setItem(row, col, cellinfo)  col += 1  row += 1 | def get\_table(self):  if self.table\_name:  self.cursor.execute(self.sql\_command["select"].format('\*', self.table\_name))  self.table = []  for i in self.cursor:  self.table.append(i)  self.attribute\_list()  self.update\_table()  def showDlgCreateDB(self):  self.dlg = QtWidgets.QDialog()  self.dlg\_ui = Create.Ui\_Dialog()  self.dlg\_ui.setupUi(self.dlg)  self.dlg\_ui.buttonBox.accepted.connect(self.create\_db)  self.dlg\_ui.buttonBox.rejected.connect(self.dlg.close)  self.dlg\_ui.tableWidget.setHorizontalHeaderLabels(('Name atribute', 'Type'))  for row in range(self.dlg\_ui.tableWidget.rowCount()):  combo = QtWidgets.QComboBox(self.dlg)  combo.addItem("INT")  combo.addItem("REAL")  combo.addItem("TEXT")  combo.addItem("TIME")  combo.addItem("DATE")  self.dlg\_ui.tableWidget.setCellWidget(row, 1, combo)  self.dlg.show()  def create\_db(self):  self.dlg.close()  try:  data = ""  for i in range(self.dlg\_ui.tableWidget.rowCount()):  if self.dlg\_ui.tableWidget.item(i, 0):  data += "{} {},".format(self.dlg\_ui.tableWidget.takeItem(i, 0).text(),  self.dlg\_ui.tableWidget.cellWidget(i, 1).currentText())  self.table\_name = self.dlg\_ui.lineEdit.text()  self.cursor.execute(self.sql\_command["create"].format(self.table\_name, data[:-1]))  self.table\_names.append(self.table\_name)  self.update\_comboBox()  except:  QMessageBox.critical(self, "Error", "Unknown error")  def showDlgCortegeAdd(self):  self.dlg = QtWidgets.QDialog()  self.dlg\_ui = Cortege.Ui\_Dialog()  self.dlg\_ui.setupUi(self.dlg)  self.dlg\_ui.tableWidget.setColumnCount(len(self.table\_header))  self.dlg\_ui.tableWidget.setHorizontalHeaderLabels(self.table\_header)  self.dlg.show()  self.dlg\_ui.buttonBox.accepted.connect(self.insert\_cortege\_db)  self.dlg\_ui.buttonBox.rejected.connect(self.dlg.close)  def insert\_cortege\_db(self):  try:  text = ""  for row in range(self.dlg\_ui.tableWidget.rowCount()):  for col in range(len(self.table\_header)):  if self.dlg\_ui.tableWidget.item(row, col):  text += self.dlg\_ui.tableWidget.item(row, col).text() + ','  else:  text += "NULL" + ','  if text:  insert = sql.SQL(self.sql\_command["insert"].format(self.table\_name, text[:-1]))  self.cursor.execute(insert)  self.get\_table()  except:  QMessageBox.critical(self, "Error", "Unknown error")  def delete\_cortege\_db(self):  try:  text, ok = QInputDialog.getText(self, "Drop", "Which cortege to drop: ")  if text and ok:  insert = sql.SQL(self.sql\_command["delete"].format(self.table\_name, text))  self.cursor.execute(insert)  self.get\_table()  except:  QMessageBox.critical(self, "Error", "Unknown error")  def showDlgAttributeAdd(self):  self.dlga = QtWidgets.QDialog()  self.dlga\_ui = Attribute.Ui\_Dialog()  self.dlga\_ui.setupUi(self.dlga)  self.dlg\_ui.tableWidget.setColumnCount(len(self.table\_header))  self.dlg\_ui.tableWidget.setHorizontalHeaderLabels(self.table\_header)  self.dlga.show()  self.dlga\_ui.buttonBox.accepted.connect(self.insert\_attribute\_db)  self.dlga\_ui.buttonBox.rejected.connect(self.dlga.close)  def insert\_attribute\_db(self):  try:  text = self.dlga\_ui.lineEdit.text()  type1 = self.dlga\_ui.comboBox.currentText()  if text:  insert = sql.SQL(self.sql\_command["addcol"].format(self.table\_name, text,type1))  self.cursor.execute(insert)  self.get\_table()  except:  QMessageBox.critical(self, "Error", "Unknown error")  def delete\_attribute\_db(self):  try:  text, ok = QInputDialog.getText(self, "Drop", "Which attribute to drop: ")  if text and ok:  insert = sql.SQL(self.sql\_command["dropcol"].format(self.table\_name, text))  self.cursor.execute(insert)  self.get\_table()  except:  QMessageBox.critical(self, "Error", "Unknown error")  def delete\_db(self):  try:  if self.table\_names:  self.table\_name, ok = QInputDialog.getItem(self, "Select table",  "Name table:", self.table\_names, 0, False)  if ok and self.table\_name:  self.cursor.execute(self.sql\_command["drop"].format(self.table\_name))  self.table\_names.remove(self.table\_name)  self.update\_comboBox()  except:  QMessageBox.critical(self, "Error", "Unknown error")  def init\_comboBox(self):  self.table\_names = []  for name in self.cursor:  self.table\_names.append(name[0])  self.update\_comboBox()  if self.table\_names:  self.table\_name = self.table\_names[0]  self.get\_table()  def update\_comboBox(self):  self.ui.comboBox.clear()  for name in self.table\_names:  self.ui.comboBox.addItem(name)  self.ui.comboBox.currentTextChanged.connect(self.update\_table\_name)  def save(self):  try:  with open(self.fname, "w", errors='ignore', encoding='utf-8', newline='') as f:  writer = csv.writer(f, delimiter=';')  writer.writerow(self.table\_header)  for row in self.table:  writer.writerow(row)  except IOError:  QMessageBox.critical(self, "Error", "File not save")    def save\_as(self):  self.fname = QFileDialog.getSaveFileName(self, 'Save as...', '/BD')[0]  try:  with open(self.fname, "w", errors='ignore', encoding='utf-8', newline='') as f:  writer = csv.writer(f, delimiter=';')  writer.writerow(self.table\_header)  for row in self.table:  writer.writerow(row)  except IOError:  QMessageBox.critical(self, "Error", "File not save")  def close\_app(self):  self.conn.close()  self.close()  def main():  app = QtWidgets.QApplication([])  application = DB\_GUI()  application.show()  sys.exit(app.exec())  if \_\_name\_\_ == "\_\_main\_\_":  main() |

В данном файле описываются все функции приложения, запросы в БД и так далее, не считая дизайна. Теперь приложим листинг всех файлов дизайна.  
  
Файл DataBaseGUI.py

|  |  |
| --- | --- |
| from PyQt5 import QtCore, QtGui, QtWidgets  class Ui\_MainWindow(object):  def setupUi(self, MainWindow):  MainWindow.setObjectName("MainWindow")  MainWindow.resize(1107, 845)  MainWindow.setStyleSheet("background-color: rgb(238, 238, 236);")  self.centralwidget = QtWidgets.QWidget(MainWindow)  self.centralwidget.setObjectName("centralwidget")  self.tableWidget = QtWidgets.QTableWidget(self.centralwidget)  self.tableWidget.setGeometry(QtCore.QRect(10, 70, 1081, 721))  self.tableWidget.setObjectName("tableWidget")  self.tableWidget.setColumnCount(0)  self.tableWidget.setRowCount(0)  self.comboBox = QtWidgets.QComboBox(self.centralwidget)  self.comboBox.setGeometry(QtCore.QRect(930, 30, 151, 25))  self.comboBox.setObjectName("comboBox")  self.label = QtWidgets.QLabel(self.centralwidget)  self.label.setGeometry(QtCore.QRect(830, 30, 71, 21))  self.label.setObjectName("label")  self.label\_2 = QtWidgets.QLabel(self.centralwidget)  self.label\_2.setGeometry(QtCore.QRect(830, 0, 91, 21))  self.label\_2.setObjectName("label\_2")  self.label\_3 = QtWidgets.QLabel(self.centralwidget)  self.label\_3.setGeometry(QtCore.QRect(960, 0, 121, 21))  self.label\_3.setText("")  self.label\_3.setObjectName("label\_3")  self.pushButton = QtWidgets.QPushButton(self.centralwidget)  self.pushButton.setGeometry(QtCore.QRect(10, 10, 231, 51))  self.pushButton.setObjectName("pushButton")  ##  self.pushButton2 = QtWidgets.QPushButton(self.centralwidget)  self.pushButton2.setGeometry(QtCore.QRect(300, 10, 231, 51))  self.pushButton2.setObjectName("pushButton2")  ##  MainWindow.setCentralWidget(self.centralwidget)  self.menubar = QtWidgets.QMenuBar(MainWindow)  self.menubar.setGeometry(QtCore.QRect(0, 0, 1107, 26))  self.menubar.setObjectName("menubar")  self.menu = QtWidgets.QMenu(self.menubar)  self.menu.setObjectName("menu")  self.menu\_2 = QtWidgets.QMenu(self.menubar)  self.menu\_2.setObjectName("menu\_2")  self.menu\_3 = QtWidgets.QMenu(self.menu\_2)  self.menu\_3.setObjectName("menu\_3")  self.menu\_4 = QtWidgets.QMenu(self.menu\_2)  self.menu\_4.setObjectName("menu\_4")  MainWindow.setMenuBar(self.menubar)  self.statusbar = QtWidgets.QStatusBar(MainWindow)  self.statusbar.setObjectName("statusbar")  MainWindow.setStatusBar(self.statusbar)  self.action = QtWidgets.QAction(MainWindow)  self.action.setObjectName("action")  self.action\_2 = QtWidgets.QAction(MainWindow)  self.action\_2.setObjectName("action\_2")  self.action\_3 = QtWidgets.QAction(MainWindow)  self.action\_3.setObjectName("action\_3")  self.action\_4 = QtWidgets.QAction(MainWindow)  self.action\_4.setObjectName("action\_4")  self.action\_5 = QtWidgets.QAction(MainWindow)  self.action\_5.setObjectName("action\_5")  self.action\_7 = QtWidgets.QAction(MainWindow)  self.action\_7.setObjectName("action\_7") | self.action\_9 = QtWidgets.QAction(MainWindow)  self.action\_9.setObjectName("action\_9")  self.action\_10 = QtWidgets.QAction(MainWindow)  self.action\_10.setObjectName("action\_10")  self.action\_6 = QtWidgets.QAction(MainWindow)  self.action\_6.setObjectName("action\_6")  self.action\_11 = QtWidgets.QAction(MainWindow)  self.action\_11.setObjectName("action\_11")  self.action\_12 = QtWidgets.QAction(MainWindow)  self.action\_12.setObjectName("action\_12")  self.action\_13 = QtWidgets.QAction(MainWindow)  self.action\_13.setObjectName("action\_13")  self.action\_8 = QtWidgets.QAction(MainWindow)  self.action\_8.setObjectName("action\_8")  self.actionPrimary\_key = QtWidgets.QAction(MainWindow)  self.actionPrimary\_key.setObjectName("actionPrimary\_key")  self.actionPrimary\_key\_2 = QtWidgets.QAction(MainWindow)  self.actionPrimary\_key\_2.setObjectName("actionPrimary\_key\_2")  self.actionAttribute = QtWidgets.QAction(MainWindow)  self.actionAttribute.setObjectName("actionAttribute")  self.actionAttribute\_2 = QtWidgets.QAction(MainWindow)  self.actionAttribute\_2.setObjectName("actionAttribute\_2")  self.menu.addAction(self.action\_2)  self.menu.addAction(self.action\_3)  self.menu.addAction(self.action\_4)  self.menu\_3.addAction(self.action\_5)  self.menu\_3.addAction(self.action\_10)  self.menu\_3.addAction(self.actionAttribute)  self.menu\_4.addAction(self.action\_6)  self.menu\_4.addAction(self.action\_11)  self.menu\_4.addAction(self.actionAttribute\_2)  self.menu\_2.addAction(self.action\_9)  self.menu\_2.addAction(self.menu\_3.menuAction())  self.menu\_2.addAction(self.menu\_4.menuAction())  self.menubar.addAction(self.menu.menuAction())  self.menubar.addAction(self.menu\_2.menuAction())  self.retranslateUi(MainWindow)  QtCore.QMetaObject.connectSlotsByName(MainWindow)  def retranslateUi(self, MainWindow):  \_translate = QtCore.QCoreApplication.translate  MainWindow.setWindowTitle(\_translate("MainWindow", "DataBase Managemen System (DBMS)"))  self.label.setText(\_translate("MainWindow", "ТАБЛИЦА:"))  self.label\_2.setText(\_translate("MainWindow", "БАЗА ДАННЫХ:"))  self.pushButton.setText(\_translate("MainWindow", "Изменить кортеж"))  self.pushButton2.setText(\_translate("MainWindow", "Показать"))  self.menu.setTitle(\_translate("MainWindow", "Файл"))  self.menu\_2.setTitle(\_translate("MainWindow", "База данных"))  self.menu\_3.setTitle(\_translate("MainWindow", "Добавить"))  self.menu\_4.setTitle(\_translate("MainWindow", "Удалить"))  self.action.setText(\_translate("MainWindow", "Открыть"))  self.action\_2.setText(\_translate("MainWindow", "Сохранить"))  self.action\_3.setText(\_translate("MainWindow", "Сохранить как"))  self.action\_4.setText(\_translate("MainWindow", "Выход"))  self.action\_5.setText(\_translate("MainWindow", "Таблица"))  self.action\_7.setText(\_translate("MainWindow", "Атрибут"))  self.action\_9.setText(\_translate("MainWindow", "Подключение"))  self.action\_10.setText(\_translate("MainWindow", "Кортеж"))  self.action\_6.setText(\_translate("MainWindow", "Таблица"))  self.action\_11.setText(\_translate("MainWindow", "Кортеж"))  self.action\_12.setText(\_translate("MainWindow", "Атрибут"))  self.action\_13.setText(\_translate("MainWindow", "Список таблиц"))  self.action\_8.setText(\_translate("MainWindow", "Скопировать"))  self.actionPrimary\_key.setText(\_translate("MainWindow", "Primary key"))  self.actionPrimary\_key\_2.setText(\_translate("MainWindow", "Primary key"))  self.actionAttribute.setText(\_translate("MainWindow", "Атрибут"))  self.actionAttribute\_2.setText(\_translate("MainWindow", "Атрибут")) |

Здесь содержится дизайн главного окна приложения, меню, размеры окна, названия, кнопки и так далее.

Файл Attribute.py

|  |  |
| --- | --- |
| from PyQt5 import QtCore, QtGui, QtWidgets  class Ui\_Dialog(object):  def setupUi(self, Dialog):  Dialog.setObjectName("Dialog")  Dialog.resize(300, 137)  self.buttonBox = QtWidgets.QDialogButtonBox(Dialog)  self.buttonBox.setGeometry(QtCore.QRect(-60, 90, 341, 32))  self.buttonBox.setOrientation(QtCore.Qt.Horizontal)  self.buttonBox.setStandardButtons(QtWidgets.QDialogButtonBox.Cancel|QtWidgets.QDialogButtonBox.Ok)  self.buttonBox.setObjectName("buttonBox")  self.comboBox = QtWidgets.QComboBox(Dialog)  self.comboBox.setGeometry(QtCore.QRect(190, 40, 91, 31))  self.comboBox.setObjectName("comboBox")  self.comboBox.addItem("INT")  self.comboBox.addItem("REAL")  self.comboBox.addItem("TEXT")  self.comboBox.addItem("TIME")  self.comboBox.addItem("DATE")  self.lineEdit = QtWidgets.QLineEdit(Dialog) | from PyQt5 import QtCore, QtGui, QtWidgets  class Ui\_Dialog(object):  def setupUi(self, Dialog):  Dialog.setObjectName("Dialog")  Dialog.resize(300, 137)  self.buttonBox = QtWidgets.QDialogButtonBox(Dialog)  self.buttonBox.setGeometry(QtCore.QRect(-60, 90, 341, 32))  self.buttonBox.setOrientation(QtCore.Qt.Horizontal)  self.buttonBox.setStandardButtons(QtWidgets.QDialogButtonBox.Cancel|QtWidgets.QDialogButtonBox.Ok)  self.buttonBox.setObjectName("buttonBox")  self.comboBox = QtWidgets.QComboBox(Dialog)  self.comboBox.setGeometry(QtCore.QRect(190, 40, 91, 31))  self.comboBox.setObjectName("comboBox")  self.comboBox.addItem("INT")  self.comboBox.addItem("REAL")  self.comboBox.addItem("TEXT")  self.comboBox.addItem("TIME")  self.comboBox.addItem("DATE")  self.lineEdit = QtWidgets.QLineEdit(Dialog) |

Файл Change.py

|  |  |
| --- | --- |
| from PyQt5 import QtCore, QtGui, QtWidgets  class Ui\_Dialog(object):  def setupUi(self, Dialog):  Dialog.setObjectName("Dialog")  Dialog.resize(400, 300)  self.buttonBox = QtWidgets.QDialogButtonBox(Dialog)  self.buttonBox.setGeometry(QtCore.QRect(30, 240, 341, 32))  self.buttonBox.setOrientation(QtCore.Qt.Horizontal)  self.buttonBox.setStandardButtons(QtWidgets.QDialogButtonBox.Cancel|QtWidgets.QDialogButtonBox.Ok)  self.buttonBox.setObjectName("buttonBox")  self.lineEdit = QtWidgets.QLineEdit(Dialog)  self.lineEdit.setGeometry(QtCore.QRect(30, 50, 91, 31))  self.lineEdit.setObjectName("lineEdit")  self.lineEdit\_2 = QtWidgets.QLineEdit(Dialog)  self.lineEdit\_2.setGeometry(QtCore.QRect(220, 50, 121, 31))  self.lineEdit\_2.setObjectName("lineEdit\_2")  self.lineEdit\_3 = QtWidgets.QLineEdit(Dialog)  self.lineEdit\_3.setGeometry(QtCore.QRect(40, 160, 341, 41))  self.lineEdit\_3.setObjectName("lineEdit\_3")  self.label = QtWidgets.QLabel(Dialog)  self.label.setGeometry(QtCore.QRect(20, 10, 131, 31))  self.label.setObjectName("label")  self.label\_2 = QtWidgets.QLabel(Dialog)  self.label\_2.setGeometry(QtCore.QRect(220, 10, 171, 31))  self.label\_2.setObjectName("label\_2")  self.label\_3 = QtWidgets.QLabel(Dialog)  self.label\_3.setGeometry(QtCore.QRect(160, 110, 261, 51))  font = QtGui.QFont() | font.setPointSize(10)  font.setBold(True)  font.setWeight(75)  self.label\_3.setFont(font)  self.label\_3.setObjectName("label\_3")  self.retranslateUi(Dialog)  self.buttonBox.accepted.connect(Dialog.accept)  self.buttonBox.rejected.connect(Dialog.reject)  QtCore.QMetaObject.connectSlotsByName(Dialog)  def retranslateUi(self, Dialog):  \_translate = QtCore.QCoreApplication.translate  Dialog.setWindowTitle(\_translate("Dialog", "Изменение строки"))  self.label.setText(\_translate("Dialog", "Изменяемый столбец"))  self.label\_2.setText(\_translate("Dialog", "Изменить при каком условии?"))  self.label\_3.setText(\_translate("Dialog", "Новое значение")) |

Файл Connection.py

|  |  |
| --- | --- |
| from PyQt5 import QtCore, QtGui, QtWidgets  class Ui\_Dialog(object):  def setupUi(self, Dialog):  Dialog.setObjectName("Dialog")  Dialog.resize(325, 218)  Dialog.setWindowIcon(QtGui.QIcon("image/database.png"))  self.horizontalLayout\_5 = QtWidgets.QHBoxLayout(Dialog)  self.horizontalLayout\_5.setObjectName("horizontalLayout\_5")  self.gridLayout\_2 = QtWidgets.QGridLayout()  self.gridLayout\_2.setObjectName("gridLayout\_2")  self.gridLayout = QtWidgets.QGridLayout()  self.gridLayout.setObjectName("gridLayout")  self.horizontalLayout = QtWidgets.QHBoxLayout()  self.horizontalLayout.setObjectName("horizontalLayout")  self.label = QtWidgets.QLabel(Dialog)  self.label.setObjectName("label")  self.horizontalLayout.addWidget(self.label)  spacerItem = QtWidgets.QSpacerItem(15, 20, QtWidgets.QSizePolicy.Expanding, QtWidgets.QSizePolicy.Minimum)  self.horizontalLayout.addItem(spacerItem)  self.lineEdit = QtWidgets.QLineEdit(Dialog)  self.lineEdit.setPlaceholderText("")  self.lineEdit.setObjectName("lineEdit")  self.horizontalLayout.addWidget(self.lineEdit)  self.gridLayout.addLayout(self.horizontalLayout, 0, 0, 1, 1)  self.horizontalLayout\_2 = QtWidgets.QHBoxLayout()  self.horizontalLayout\_2.setObjectName("horizontalLayout\_2")  self.label\_2 = QtWidgets.QLabel(Dialog)  self.label\_2.setObjectName("label\_2")  self.horizontalLayout\_2.addWidget(self.label\_2)  spacerItem1 = QtWidgets.QSpacerItem(15, 20, QtWidgets.QSizePolicy.Expanding, QtWidgets.QSizePolicy.Minimum)  self.horizontalLayout\_2.addItem(spacerItem1)  self.lineEdit\_2 = QtWidgets.QLineEdit(Dialog) | self.lineEdit\_2.setObjectName("lineEdit\_2")  self.horizontalLayout\_2.addWidget(self.lineEdit\_2)  self.gridLayout.addLayout(self.horizontalLayout\_2, 1, 0, 1, 1)  self.horizontalLayout\_3 = QtWidgets.QHBoxLayout()  self.horizontalLayout\_3.setObjectName("horizontalLayout\_3")  self.label\_3 = QtWidgets.QLabel(Dialog)  self.label\_3.setObjectName("label\_3")  self.horizontalLayout\_3.addWidget(self.label\_3)  spacerItem2 = QtWidgets.QSpacerItem(40, 20, QtWidgets.QSizePolicy.Expanding, QtWidgets.QSizePolicy.Minimum)  self.horizontalLayout\_3.addItem(spacerItem2)  self.lineEdit\_3 = QtWidgets.QLineEdit(Dialog)  self.lineEdit\_3.setEchoMode(QtWidgets.QLineEdit.Password)  self.lineEdit\_3.setObjectName("lineEdit\_3")  self.horizontalLayout\_3.addWidget(self.lineEdit\_3)  self.gridLayout.addLayout(self.horizontalLayout\_3, 2, 0, 1, 1)  self.horizontalLayout\_4 = QtWidgets.QHBoxLayout()  self.horizontalLayout\_4.setObjectName("horizontalLayout\_4")  self.label\_4 = QtWidgets.QLabel(Dialog)  self.label\_4.setObjectName("label\_4")  self.horizontalLayout\_4.addWidget(self.label\_4)  spacerItem3 = QtWidgets.QSpacerItem(40, 20, QtWidgets.QSizePolicy.Expanding, QtWidgets.QSizePolicy.Minimum)  self.horizontalLayout\_4.addItem(spacerItem3)  self.lineEdit\_4 = QtWidgets.QLineEdit(Dialog)  self.lineEdit\_4.setObjectName("lineEdit\_4")  self.horizontalLayout\_4.addWidget(self.lineEdit\_4)  self.gridLayout.addLayout(self.horizontalLayout\_4, 3, 0, 1, 1)  self.gridLayout\_2.addLayout(self.gridLayout, 0, 0, 1, 1)  self.buttonBox = QtWidgets.QDialogButtonBox(Dialog)  self.buttonBox.setOrientation(QtCore.Qt.Horizontal)  self.buttonBox.setStandardButtons(QtWidgets.QDialogButtonBox.Cancel|QtWidgets.QDialogButtonBox.Ok)  self.buttonBox.setObjectName("buttonBox")  self.gridLayout\_2.addWidget(self.buttonBox, 1, 0, 1, 1)  self.horizontalLayout\_5.addLayout(self.gridLayout\_2)  self.retranslateUi(Dialog)  self.buttonBox.accepted.connect(Dialog.accept)  self.buttonBox.rejected.connect(Dialog.reject)  QtCore.QMetaObject.connectSlotsByName(Dialog)  def retranslateUi(self, Dialog):  \_translate = QtCore.QCoreApplication.translate  Dialog.setWindowTitle(\_translate("Dialog", "Connect to Data Base"))  self.label.setText(\_translate("Dialog", "DataBase\_name"))  self.lineEdit.setText(\_translate("Dialog", "postgres"))  self.label\_2.setText(\_translate("Dialog", "Server name"))  self.lineEdit\_2.setText(\_translate("Dialog", "postgres"))  self.label\_3.setText(\_translate("Dialog", "Password"))  self.lineEdit\_3.setText(\_translate("Dialog", "123456"))  self.label\_4.setText(\_translate("Dialog", "IP host"))  self.lineEdit\_4.setText(\_translate("Dialog", "localhost")) |

Файл Cortege.py

|  |  |
| --- | --- |
| from PyQt5 import QtCore, QtGui, QtWidgets  class Ui\_Dialog(object):  def setupUi(self, Dialog):  Dialog.setObjectName("Dialog")  Dialog.resize(296, 121)  #Dialog.setWindowIcon(QtGui.QIcon("image\\data.png"))  self.gridLayout = QtWidgets.QGridLayout(Dialog)  self.gridLayout.setObjectName("gridLayout")  self.verticalLayout = QtWidgets.QVBoxLayout()  self.verticalLayout.setObjectName("verticalLayout")  self.tableWidget = QtWidgets.QTableWidget(Dialog)  self.tableWidget.setRowCount(1)  self.tableWidget.setObjectName("tableWidget")  self.tableWidget.setColumnCount(0) | self.verticalLayout.addWidget(self.tableWidget)  self.buttonBox = QtWidgets.QDialogButtonBox(Dialog)  self.buttonBox.setOrientation(QtCore.Qt.Horizontal)  self.buttonBox.setStandardButtons(QtWidgets.QDialogButtonBox.Cancel|QtWidgets.QDialogButtonBox.Ok)  self.buttonBox.setObjectName("buttonBox")  self.verticalLayout.addWidget(self.buttonBox)  self.gridLayout.addLayout(self.verticalLayout, 0, 0, 1, 1)  self.retranslateUi(Dialog)  self.buttonBox.accepted.connect(Dialog.accept)  self.buttonBox.rejected.connect(Dialog.reject)  QtCore.QMetaObject.connectSlotsByName(Dialog)  def retranslateUi(self, Dialog):  \_translate = QtCore.QCoreApplication.translate  Dialog.setWindowTitle(\_translate("Dialog", "Добавить строку")) |

Файл Create.py

|  |  |
| --- | --- |
| from PyQt5 import QtCore, QtGui, QtWidgets  class Ui\_Dialog(object):  def setupUi(self, Dialog):  Dialog.setObjectName("Dialog")  Dialog.setFixedSize(278, 318)  Dialog.setWindowIcon(QtGui.QIcon("image/insert\_table.png"))  self.buttonBox = QtWidgets.QDialogButtonBox(Dialog)  self.buttonBox.setGeometry(QtCore.QRect(70, 270, 181, 32))  self.buttonBox.setOrientation(QtCore.Qt.Horizontal)  self.buttonBox.setStandardButtons(QtWidgets.QDialogButtonBox.Cancel|QtWidgets.QDialogButtonBox.Ok)  self.buttonBox.setObjectName("buttonBox")  self.tableWidget = QtWidgets.QTableWidget(Dialog)  self.tableWidget.setGeometry(QtCore.QRect(10, 70, 241, 192))  self.tableWidget.setFrameShape(QtWidgets.QFrame.StyledPanel)  self.tableWidget.setRowCount(10)  self.tableWidget.setColumnCount(2)  self.tableWidget.setObjectName("tableWidget")  self.tableWidget.horizontalHeader().setCascadingSectionResizes(True)  self.tableWidget.horizontalHeader().setSortIndicatorShown(False)  self.horizontalLayoutWidget = QtWidgets.QWidget(Dialog) | self.horizontalLayoutWidget.setGeometry(QtCore.QRect(10, 20, 241, 41))  self.horizontalLayoutWidget.setObjectName("horizontalLayoutWidget")  self.horizontalLayout = QtWidgets.QHBoxLayout(self.horizontalLayoutWidget)  self.horizontalLayout.setContentsMargins(0, 0, 0, 0)  self.horizontalLayout.setObjectName("horizontalLayout")  self.label = QtWidgets.QLabel(self.horizontalLayoutWidget)  self.label.setObjectName("label")  self.horizontalLayout.addWidget(self.label)  self.lineEdit = QtWidgets.QLineEdit(self.horizontalLayoutWidget)  self.lineEdit.setObjectName("lineEdit")  self.horizontalLayout.addWidget(self.lineEdit)  self.retranslateUi(Dialog)  self.buttonBox.accepted.connect(Dialog.accept)  self.buttonBox.rejected.connect(Dialog.reject)  QtCore.QMetaObject.connectSlotsByName(Dialog)  def retranslateUi(self, Dialog):  \_translate = QtCore.QCoreApplication.translate  Dialog.setWindowTitle(\_translate("Dialog", "Создание таблицы"))  self.tableWidget.setSortingEnabled(False)  self.label.setText(\_translate("Dialog", "Название таблицы")) |

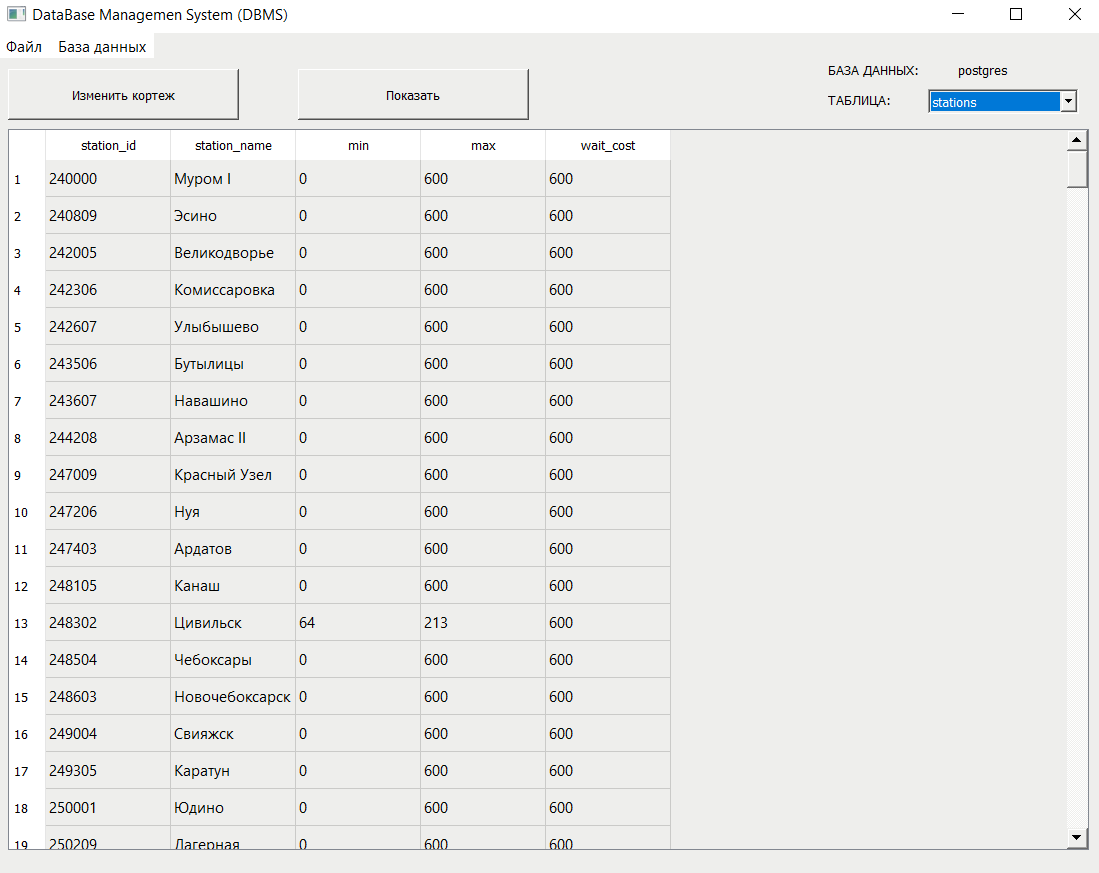
Файл Select.py

|  |  |
| --- | --- |
| from PyQt5 import QtCore, QtGui, QtWidgets  class Ui\_Dialog(object):  def setupUi(self, Dialog):  Dialog.setObjectName("Dialog")  Dialog.resize(826, 447)  self.buttonBox = QtWidgets.QDialogButtonBox(Dialog)  self.buttonBox.setGeometry(QtCore.QRect(460, 400, 341, 32))  self.buttonBox.setOrientation(QtCore.Qt.Horizontal)  self.buttonBox.setStandardButtons(QtWidgets.QDialogButtonBox.Cancel|QtWidgets.QDialogButtonBox.Ok)  self.buttonBox.setObjectName("buttonBox")  self.lineEdit = QtWidgets.QLineEdit(Dialog)  self.lineEdit.setGeometry(QtCore.QRect(80, 180, 651, 31))  self.lineEdit.setObjectName("lineEdit")  self.label = QtWidgets.QLabel(Dialog)  self.label.setGeometry(QtCore.QRect(300, 150, 231, 16))  self.label.setObjectName("label")  self.lineEdit\_2 = QtWidgets.QLineEdit(Dialog)  self.lineEdit\_2.setGeometry(QtCore.QRect(80, 60, 651, 31)) | self.lineEdit\_2.setObjectName("lineEdit\_2")  self.label\_2 = QtWidgets.QLabel(Dialog)  self.label\_2.setGeometry(QtCore.QRect(280, 30, 281, 16))  self.label\_2.setObjectName("label\_2")  self.retranslateUi(Dialog)  self.buttonBox.accepted.connect(Dialog.accept)  self.buttonBox.rejected.connect(Dialog.reject)  QtCore.QMetaObject.connectSlotsByName(Dialog)  def retranslateUi(self, Dialog):  \_translate = QtCore.QCoreApplication.translate  Dialog.setWindowTitle(\_translate("Dialog", "Dialog"))  self.label.setText(\_translate("Dialog", "Условие вывода (\"при условии\")"))  self.label\_2.setText(\_translate("Dialog", "Какие столбцы вывести (через запятую)")) |

Файл Select\_Table.py

|  |  |
| --- | --- |
| from PyQt5 import QtCore, QtGui, QtWidgets  class Ui\_Dialog(object):  def setupUi(self, Dialog):  Dialog.setObjectName("Dialog")  Dialog.resize(850, 753)  self.buttonBox = QtWidgets.QDialogButtonBox(Dialog)  self.buttonBox.setGeometry(QtCore.QRect(190, 690, 341, 32))  self.buttonBox.setOrientation(QtCore.Qt.Horizontal)  self.buttonBox.setStandardButtons(QtWidgets.QDialogButtonBox.Cancel|QtWidgets.QDialogButtonBox.Ok)  self.buttonBox.setObjectName("buttonBox")  self.tableWidget = QtWidgets.QTableWidget(Dialog) | self.tableWidget.setGeometry(QtCore.QRect(80, 60, 711, 521))  self.tableWidget.setObjectName("tableWidget")  self.tableWidget.setColumnCount(0)  self.tableWidget.setRowCount(0)  self.retranslateUi(Dialog)  self.buttonBox.accepted.connect(Dialog.accept)  self.buttonBox.rejected.connect(Dialog.reject)  QtCore.QMetaObject.connectSlotsByName(Dialog)  def retranslateUi(self, Dialog):  \_translate = QtCore.QCoreApplication.translate  Dialog.setWindowTitle(\_translate("Dialog", "Вывод")) |

Внешний вид главного окна при подключенной базе данных:



# **ВЫВОД**

В ходе проделанной работы были дополнительно закреплены основы работы с объектно-реляционной системой управления базами данных PostgreSQL, языком Python и SQL. Было написано приложение для работы с базами данных для людей, не владеющих знаниями в данных областях. Были получены навыки по интеграции баз данных в языке Python, а также по работе с оболочкой PyQt5, значительно упрощающей написание дизайна окон программ. Также были подробно рассмотрены различные структуры баз данных.

Получившееся приложение можно использовать в различных сферах, в качестве упрощенной СУБД для работников педагогической отрасли (для выставления оценок, баллов и так далее), для специалистов в ведении бухгалтерского учета, работников в сфере аудита и консалтинга и многих других.

# **СПИСОК ИСПОЛЬЗОВАННОЙ ЛИТЕРАТУРЫ**

1. Карпова Т.С. Базы данных: модели, разработка, реализация: учебник для вузов. СПб.: Питер, 2001. - 303 с.
2. Прохоренок Н.А. Python 3 и PyQt 5. Разработка приложений: Пособие. СПб:БХВ-Петербург, 2016. - 833 с.
3. Шкарина Л. Язык SQL: учеб. курс. СПб.: Питер, 2001. - 592 с.
4. Марков А.С., Лисовский К.Ю. Базы данных. Введение в теорию и методологию: учебник. М.: Финансы и статистика, 2006. - 512 с.