Федеральное государственное бюджетное образовательное учреждение высшего профессионального образования

«**КАЛИНИНГРАДСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ**»

Кафедра систем управления и вычислительной техники

|  |  |
| --- | --- |
|  | Отчет защищен с оценкой  …………………  Руководитель доц. Высоцкий Л.Г. |

………………………

Дата \_\_\_\_\_\_\_\_\_\_\_\_\_\_

**ОТЧЕТ**

по учебной (ознакомительной) практике

ст. гр. 19-ИЭ

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Дата \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Оглавление**

[1) Вариант задания на выполнение проекта 3](#_Toc76634145)

[2) Структура проекта 3](#_Toc76634146)

[3) Структура таблиц 4](#_Toc76634147)

[4) Тексты всех запросов 4](#_Toc76634148)

[5) Листинг программы 6](#_Toc76634149)

[Список литературы: 17](#_Toc76634150)

# **Вариант задания на выполнение проекта**

1. В соответствии с вариантом из таблицы выбирается предметная область будущей БД.

2. На основе заданной предметной области разрабатывается структура двух таб-лиц, аналогично рис. 3. В 1-й таблице должно быть не менее 4-х полей, во 2-й – не менее 3-х полей. В обеих таблицах обязательны ключевые поля, причем клю-чевое поле 2-й таблицы должно совпадать с одним из неключевых полей 1-й таблицы.

В 1-й таблице минимум одно поле должно быть числовым, одно текстовым, одно типа даты и/или времени. Для полей, где это возможно, должны быть установле-ны ограничения.

3. Для 1-й таблицы разрабатывается программный интерфейс, предполагающий четыре режима доступа: просмотр, дополнение, модификация, удаление. Режим задается компонентом, указанным для варианта в таблице

4. 1-я таблица заполняется не менее чем 20-ю записями, 2-я – не менее трех любым способом.

5. Создаются три запроса выбора для БД:

а) параметризованный запрос типа (2);

б) параметризованный запрос с группировкой по одному из неключевых полей типа (1);

в) сложный запрос по двум таблицам типа (3).

|  |  |  |  |
| --- | --- | --- | --- |
| **№ варианта** | **Предметная область** | **Переключение режимов работы с таблицей** | **Выбор вида запроса** |
| 3 | Музыкальные инструменты | Ниспадающее меню | Главное меню |

# **Структура проекта**

* 1. 3 виджета Treeview для отображения данных из БД
  2. Ниспадающее меню для переключения режимов работы с таблицей
  3. Главное меню из четырёх пунктов для выбора вида запроса

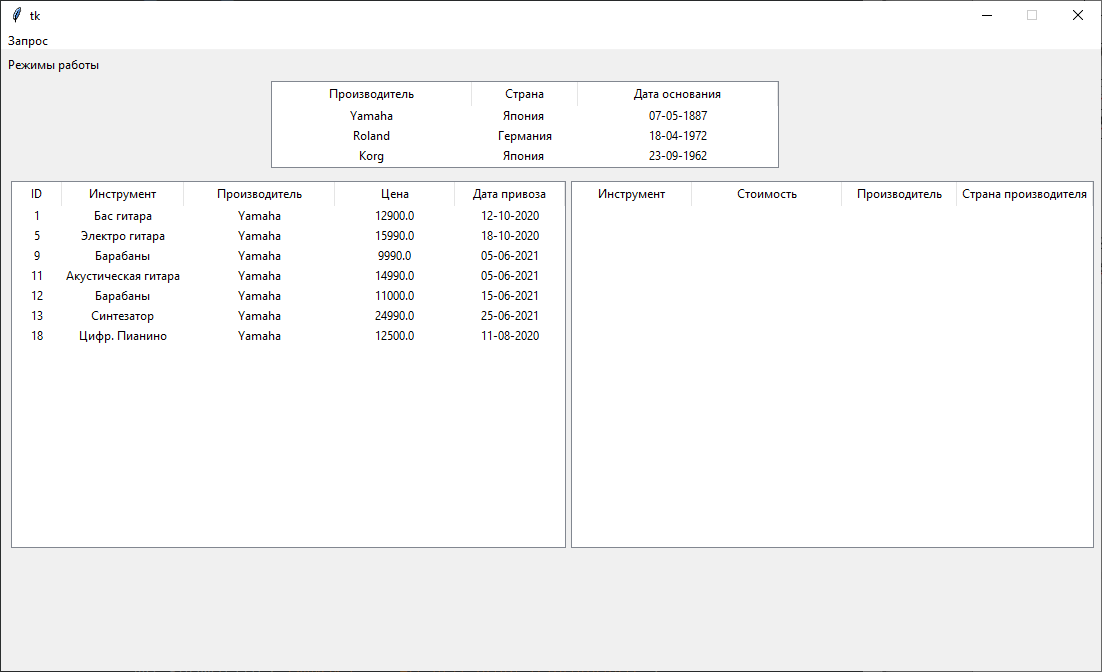
# **Структура таблиц**

|  |  |  |
| --- | --- | --- |
| **Instrument** | | |
| *Имя* | *Тип* | *Ограничения* |
| Drummers_keyID (Номер) | INT | >0 |
| instrument\_name | TEXT | 20 |
| instrument\_Company | TEXT | 20 |
| Instrument\_Price | Real | 20 |
| date\_privoza | timestamp | <2022 and >1980 |

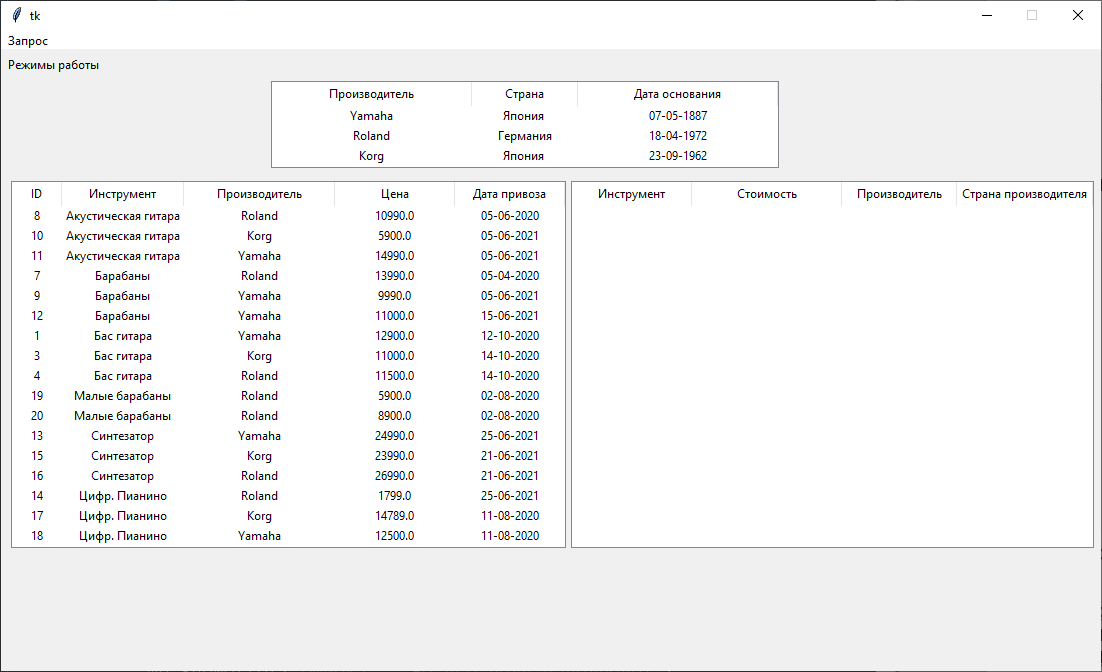
|  |  |  |
| --- | --- | --- |
| **CompanyTable** | | |
| *Имя* | *Тип* | *Ограничения* |
| Drummers_key Company\_Name | TEXT | 20 |
| Company\_Strana | TEXT | 20 |
| Date\_Founded | TEXT | <2022 and >1980 |

# **Тексты всех запросов**

Параметризированный запрос

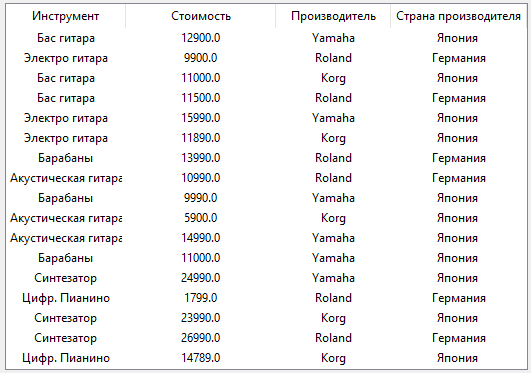
'''SELECT \* FROM Instrument WHERE instrument\_Company LIKE ?''', instrument\_Company 

Запрос с группировкой

'''SELECT \* FROM Instrument ORDER BY instrument\_Name''' 

Сложный запрос по двум таблицам

'''SELECT Instrument.instrument\_name, Instrument.instrument\_Price, CompanyTable.Company\_Name, CompanyTable.Company\_Strana FROM Instrument INNER JOIN CompanyTable ON Instrument.instrument\_Company=CompanyTable.Company\_Name '''



# **Листинг программы**

import tkinter as tk

from tkinter import ttk

from tkinter import messagebox as mb

import sqlite3

class Main(tk.Frame):

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

super().\_\_init\_\_(root)

self.init\_main()

self.db = db

self.otobrazit\_zapisi()

def init\_main(self):

self.mainmenu = tk.Menu(root)

root.config(menu=self.mainmenu)

self.searchmenu = tk.Menu(self.mainmenu, tearoff=0)

self.searchmenu.add\_command(label="Параметризированный запрос", command=self.open\_search\_dialog)

self.searchmenu.add\_command(label="Парам. запрос с группировкой", command=self.poisk\_zapisi\_Group)

self.searchmenu.add\_command(label="Запрос по двум таблицам", command=self.Search\_Two\_Table)

self.searchmenu.add\_command(label="Вернуть первую таблицу", command=self.otobrazit\_zapisi)

self.mainmenu.add\_cascade(label="Запрос", menu=self.searchmenu)

self.mb = tk.Menubutton(text="Режимы работы")

self.mb.menu = tk.Menu(self.mb, tearoff=0)

self.mb["menu"] = self.mb.menu

self.mb.place(x=1, y=1)

self.mb.menu.add\_command(label="Добавить", command=self.open\_dialog\_winow)

self.mb.menu.add\_command(label="Редактировать", command=self.proverka\_vibora\_update)

self.mb.menu.add\_command(label="Удалить", command=self.proverka\_vibora\_delete)

# show=headings скрывает нулевую колонку

self.tree = ttk.Treeview(columns=('ID', 'instrument\_name', 'instrument\_Company', 'Instrument\_Price', 'date\_privoza'),

height=17, show='headings')

# anchor=tk.CENTER выравнивает по центру заголовок

self.tree.column('ID', width=50, anchor=tk.CENTER)

self.tree.column('instrument\_name', width=122, anchor=tk.CENTER)

self.tree.column('instrument\_Company', width=150, anchor=tk.CENTER)

self.tree.column('Instrument\_Price', width=120, anchor=tk.CENTER)

self.tree.column('date\_privoza', width=110, anchor=tk.CENTER)

self.tree.heading('ID', text='ID')

self.tree.heading('instrument\_name', text='Инструмент')

self.tree.heading('instrument\_Company', text='Производитель')

self.tree.heading('Instrument\_Price', text='Цена')

self.tree.heading('date\_privoza', text='Дата привоза')

self.tree.place(x=10, y=130)

self.tree2 = ttk.Treeview(columns=('Company\_Name', 'Company\_Strana', 'Date\_Founded'), height=3, show='headings')

self.tree2.column('Company\_Name', width=200, anchor=tk.CENTER)

self.tree2.column('Company\_Strana', width=105, anchor=tk.CENTER)

self.tree2.column('Date\_Founded', width=200, anchor=tk.CENTER)

self.tree2.heading('Company\_Name', text='Производитель')

self.tree2.heading('Company\_Strana', text='Страна')

self.tree2.heading('Date\_Founded', text='Дата основания')

self.tree2.place(x=270, y=30)

self.tree3 = ttk.Treeview(columns=('instrument\_name', 'instrument\_Price', 'Company\_Name', 'Company\_Strana'), height=17,

show='headings')

self.tree3.column('instrument\_name', width=120, anchor=tk.CENTER)

self.tree3.column('instrument\_Price', width=150, anchor=tk.CENTER)

self.tree3.column('Company\_Name', width=115, anchor=tk.CENTER)

self.tree3.column('Company\_Strana', width=135, anchor=tk.CENTER)

self.tree3.heading('instrument\_name', text='Инструмент')

self.tree3.heading('instrument\_Price', text='Cтоимость')

self.tree3.heading('Company\_Name', text='Производитель')

self.tree3.heading('Company\_Strana', text='Страна производителя')

self.tree3.place(x=570, y=130)

self.Vibran\_Zapis = []

def proverka\_vibora\_update(self):

try:

self.Vibran\_Zapis = self.tree.item(self.tree.selection()[0], option="values")

self.Vibran\_Zapis = self.Vibran\_Zapis[1:]

Redaktirovanie\_Okno(app.Vibran\_Zapis)

except:

mb.showerror("Ошибка", "Не выбрана запись!")

def proverka\_vibora\_delete(self):

try:

self.Vibran\_Zapis = self.tree.item(self.tree.selection()[0], option="values")

self.Vibran\_Zapis = self.Vibran\_Zapis[1:]

self.Open\_Ask\_Delete()

except:

mb.showerror("Ошибка", "Не выбрана запись!")

def open\_dialog\_winow(\*args):

Vtoroe\_Okno()

def open\_search\_dialog(\*args):

Search()

def otobrazit\_zapisi(self):

self.db.c.execute('''SELECT \* FROM Instrument''')

for i in self.tree.get\_children():

self.tree.delete(i)

for row in self.db.c.fetchall():

self.tree.insert('', 'end', values=row)

self.db.c.execute('''SELECT \* FROM CompanyTable''')

for i in self.tree2.get\_children():

self.tree2.delete(i)

for row in self.db.c.fetchall():

self.tree2.insert('', 'end', values=row)

def zapisi(self, instrument\_name, instrument\_Company, Instrument\_Price, date\_privoza):

self.db.insert\_data(instrument\_name, instrument\_Company, Instrument\_Price, date\_privoza)

self.otobrazit\_zapisi()

def update\_record(self, instrument\_name, instrument\_Company, Instrument\_Price, date\_privoza):

self.db.c.execute(

'''UPDATE Instrument SET instrument\_name=?, instrument\_Company=?, Instrument\_Price=?, date\_privoza=? WHERE ID=?''',

(instrument\_name, instrument\_Company, Instrument\_Price, date\_privoza, self.tree.set(self.tree.selection()[0], '#1')))

self.db.conn.commit()

self.otobrazit\_zapisi()

def ydalit\_zapisi(self):

for selection\_item in self.tree.selection():

self.db.c.execute('''DELETE FROM Instrument WHERE id=?''', (self.tree.set(selection\_item, '#1'),))

self.db.conn.commit()

self.otobrazit\_zapisi()

def poisk\_zapisi(self, instrument\_Company):

instrument\_Company = ('%' + instrument\_Company + '%',)

self.db.c.execute('''SELECT \* FROM Instrument WHERE instrument\_Company LIKE ?''', instrument\_Company)

for i in self.tree.get\_children():

self.tree.delete(i)

for row in self.db.c.fetchall():

self.tree.insert('', 'end', values=row)

def poisk\_zapisi\_Group(self):

self.db.c.execute('''SELECT \* FROM Instrument ORDER BY instrument\_Name''')

for i in self.tree.get\_children():

self.tree.delete(i)

for row in self.db.c.fetchall():

self.tree.insert('', 'end', values=row)

def Search\_Two\_Table(self):

self.db.c.execute(

'''SELECT Instrument.instrument\_name, Instrument.instrument\_Price, CompanyTable.Company\_Name, CompanyTable.Company\_Strana FROM Instrument INNER JOIN CompanyTable ON Instrument.instrument\_Company=CompanyTable.Company\_Name ''')

for i in self.tree3.get\_children():

self.tree3.delete(i)

for row in self.db.c.fetchall():

self.tree3.insert('', 'end', values=row)

def Open\_Ask\_Delete(self):

answer = mb.askyesno(title="Вопрос", message="Вы действительно хотите удалить запись?")

if answer:

self.ydalit\_zapisi()

class Vtoroe\_Okno(tk.Toplevel): # Класс создания вспомогательного окна для добавления записей

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

super().\_\_init\_\_(root)

self.init\_Vtoroe\_Okno()

self.view = app

def init\_Vtoroe\_Okno(self):

self.title('Добавить позицию')

self.geometry('400x220+600+400')

self.resizable(False, False)

label\_instrument\_name = tk.Label(self, text='Инструмент:')

label\_instrument\_name.place(x=50, y=10)

label\_instrument\_Company = tk.Label(self, text='Производитель:')

label\_instrument\_Company.place(x=50, y=70)

label\_Instrument\_Price = tk.Label(self, text='Стоимость:')

label\_Instrument\_Price.place(x=50, y=40)

label\_data = tk.Label(self, text='Дата привоза')

label\_data.place(x=50, y=100)

self.entry\_instrument\_price = ttk.Entry(self)

self.entry\_instrument\_price.place(x=200, y=40)

self.entry\_instrument\_name = ttk.Entry(self)

self.entry\_instrument\_name.place(x=200, y=10)

self.conn = sqlite3.connect('InstrumentDB.db')

self.c = self.conn.cursor()

self.CompanyList = []

self.CompanyList = [Company\_Name[0] for Company\_Name in self.c.execute("SELECT Company\_Name FROM CompanyTable")]

self.companyComboBox = ttk.Combobox(self, state="readonly")

self.companyComboBox["values"] = self.CompanyList

self.companyComboBox.place(x=200, y=70)

self.numberAction = 1

self.entry\_data = ttk.Entry(self)

self.entry\_data.place(x=200, y=100)

btn\_cancel = ttk.Button(self, text='Закрыть', command=self.destroy)

btn\_cancel.place(x=200, y=170)

self.btn\_ok = ttk.Button(self, text='Добавить')

self.btn\_ok.place(x=100, y=170)

self.btn\_ok.bind('<Button-1>', lambda event: self.Check\_Empty(self.entry\_instrument\_name.get(),

self.companyComboBox.get(),

self.entry\_instrument\_price.get(),

self.entry\_data.get(),

self.numberAction))

self.grab\_set() # перехватывает все события происходящих в приложении

self.focus\_set() # захватывает и удерживает фокус

def Check\_Empty(self, instrument\_name, instrument\_Company, Instrument\_Price, date\_privoza, numberAction):

date = date\_privoza.split("-")

dateStr = "".join(date)

if len(instrument\_name) > 20 or len(instrument\_Company) > 20:

mb.showerror("Ошибка", "Превышен лимит символов!")

else:

try:

sym = float(Instrument\_Price)

except:

mb.showerror("Ошибка", "Неправильна введена цена!")

if not instrument\_name or not instrument\_Company or not Instrument\_Price or not date\_privoza:

mb.showerror("Ошибка", "Не все поля заполнены!")

else:

if sym > 0:

try:

int(dateStr)

if len(date) == 3:

for i in range(3):

date[i] = int(date[i])

else:

mb.showerror("Ошибка", "Неправильная введена дата!")

except:

mb.showerror("Ошибка", "Неправильна введена дата!")

if date[0] < 32 and date[1] < 13 and date[2] < 2022 and date[2] > 1980:

if numberAction == 1:

self.view.zapisi(instrument\_name, instrument\_Company, Instrument\_Price, date\_privoza)

mb.showinfo("Успех", "Запись добавлена!")

self.Destroy()

else:

self.view.update\_record(instrument\_name, instrument\_Company, Instrument\_Price, date\_privoza)

mb.showinfo("Успех", "Запись обновлена!")

self.Destroy()

else:

mb.showerror("Ошибка", "Неправильно введена дата!")

else:

mb.showerror("Ошибка", "Неправильна введена цена!")

def Destroy(self, \*args):

Vtoroe\_Okno.destroy(self)

class Redaktirovanie\_Okno(

Vtoroe\_Okno): # Класс создания окна для редактирования записей(наследует свойства от окна добавления)

def \_\_init\_\_(self, Vibran\_Zapis):

super().\_\_init\_\_()

self.init\_edit(Vibran\_Zapis)

self.view = app

def init\_edit(self, Vibran\_Zapis):

self.numberAction = 2

self.Vibran\_Zapis = Vibran\_Zapis

self.title('Редактировать позицию')

btn\_edit = ttk.Button(self, text='Редактировать')

self.entry\_instrument\_name.insert(0, Vibran\_Zapis[0])

self.entry\_instrument\_price.insert(0, Vibran\_Zapis[2])

self.companyComboBox.set(Vibran\_Zapis[1])

self.entry\_data.insert(0, Vibran\_Zapis[3])

btn\_edit.place(x=100, y=170)

btn\_edit.bind('<Button-1>', lambda event: self.Check\_Empty(self.entry\_instrument\_name.get(),

self.companyComboBox.get(),

self.entry\_instrument\_price.get(),

self.entry\_data.get(),

self.numberAction))

self.btn\_ok.destroy()

class Search(tk.Toplevel): # Окно для поиска записей

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

super().\_\_init\_\_()

self.init\_search()

self.view = app

def init\_search(self):

self.title('Поиск')

self.geometry('300x100+600+400')

self.resizable(False, False)

self.conn = sqlite3.connect('InstrumentDB.db')

self.c = self.conn.cursor()

self.CompanyList = []

self.CompanyList = [Company\_Name[0] for Company\_Name in self.c.execute("SELECT Company\_Name FROM CompanyTable")]

self.companyComboBox = ttk.Combobox(self, state="readonly")

self.companyComboBox["values"] = self.CompanyList

self.companyComboBox.place(x=115, y=20)

labelSearch = tk.Label(self, text='Компания')

labelSearch.place(x=15, y=20)

btn\_cancel = ttk.Button(self, text='Закрыть', command=self.destroy)

btn\_cancel.place(x=185, y=50)

btn\_search = ttk.Button(self, text='Поиск')

btn\_search.place(x=105, y=50)

btn\_search.bind('<Button-1>', lambda event: self.view.poisk\_zapisi(self.companyComboBox.get()))

btn\_search.bind('<Button-1>', lambda event: self.destroy(), add='+')

class DB:

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

self.conn = sqlite3.connect('InstrumentDB.db')

self.c = self.conn.cursor()

self.c.execute(

'''CREATE TABLE IF NOT EXISTS Instrument (

id integer primary key,

instrument\_name text,

instrument\_Company text,

Instrument\_Price real,

date\_privoza timestamp,

FOREIGN KEY (instrument\_Company) REFERENCES CompanyTable(Company\_Name))

''')

self.c.execute('''CREATE TABLE IF NOT EXISTS CompanyTable(

Company\_Name text primary key,

Company\_Strana text,

Date\_Founded text)

''')

self.conn.commit()

def insert\_data(self, instrument\_name, instrument\_Company, Instrument\_Price, date\_privoza):

self.c.execute(

'''INSERT INTO Instrument(instrument\_name, instrument\_Company, Instrument\_Price, date\_privoza) VALUES (?, ?, ?, ?)''',

(instrument\_name, instrument\_Company, Instrument\_Price, date\_privoza))

self.conn.commit()

root = tk.Tk()

root.geometry("1100x620")

root.resizable(False, False)

db = DB()

Main(root)

app = Main(root)

root.mainloop()

# **Список литературы:**

1. https://www.severcart.ru/blog/all/python\_sqlite3/

2. https://pyneng.readthedocs.io/ru/latest/book/25\_db/sqlite3.html

3. https://pythonru.com/osnovy/sqlite-v-python

4. https://digitology.tech/docs/python\_3/library/sqlite3.html