Federal State Budgetary Educational Institution

higher education

Ufa University of Science and Technology

Department of Computational Mathematics and Cybernetics

Laboratory work No.4

“Database design. Designing the user interface”

Made by: Kozlov I.A.

Group number: PRO 234

Checked by: Rizvanov D.A.

Ufa 2023

**Purpose of the work:** Introduction to basic database design techniques, introduction to user interface design techniques.

**Task:**

*Database Design*

1. Study additional material on the theory of database design (see the section "additional information") design the application database for storing processed data.

a. The database should contain a table with information about processed files (minimum set of columns: file name, date of processing).

b. The table of processed files should limit the main table with data by foreign key.

2. To write SQL- a script for creating the structure of the designed database (see the example SQL- script in the archive pikpo4\_python.zip).

3. Study the sample code from the archive pikpo4\_python.zip. Based on this example, implement the necessary CRUD (Create, Read, Update, Delete) operations for working with the database (see the link to the material on the basics SQL). Upload the app code to GitHub.

4. Check execution CRUD-operations on the test database (only SQLite).

*Designing the user interface*

During the laboratory work, it is necessary to describe the expected behavior of the developed system from the point of view of an external user, that is, to "design" external interactions of the future IC with the user without specifying its internal structure.

1. Define the structure of the designed user interface (visual design that is responsible for presenting information to the user; system functionality that includes a set of features for effective performance of professional activities; user interaction techniques with the system + additional system functionality), taking into account the task (lectures, recommendations Appendix 1, additional system functionality (example) see Appendix 3).

2. Define user interface styles (graphic (GUI, web- interface (WUI), object-oriented interface) based on the task (lectures, recommendations Appendix 1).

3. Decide on the placement of user interface elements (buttons, icons, drop-down lists, fields for writing text, etc.) (Lectures, recommendations Appendix 1).

4. Write Requirements for the user interface (See Appendix 2 for an example).

5. Develop the interaction of the developed program with the user: a script (you can use a sequence diagram or interaction diagram), on-screen forms, a set of hints (enumeration), and so on.

6. Get acquainted with the methodological material on the basic layout of web pages.

7. Based on item 1-5 develop (compare) basic html pages for your app, using CSS styles and HTML5 layout model (see the implementation example HTML5-pages in a folder www).

**Description of the work:**

**Data base creation**

SQLite – compact embedded relational database. The main feature of this database is – this works without using the client-server interaction scheme. SQLite provides a library for working with a file as a relational database, which allows it to be used in embedded systems, as well as for local testing and debugging of the application.

However, despite a number of advantages, SQLite it also has some limitations that should be taken into account during the development process:

• Multiple processes or threads can simultaneously read data from the same database, but writing to the database is only possible if no other requests are currently being served.

• Dynamic data typing is used, i.e. if the field is declared as "INTEGER", SQLite allows you to enter any type of value in this column (999, "abc", "123", 678.525). If the inserted value is not an integer, SQLite will try to convert it to an integer (the string " 123 "will turn into an integer 123), and the remaining values will be written"as is".

• Unavailable RIGHT and FULL OUTER JOIN. Implemented only LEFT OUTER JOIN.

• Partially implemented featuresALTER TABLE (available only RENAME TABLE and ADD COLUMN).

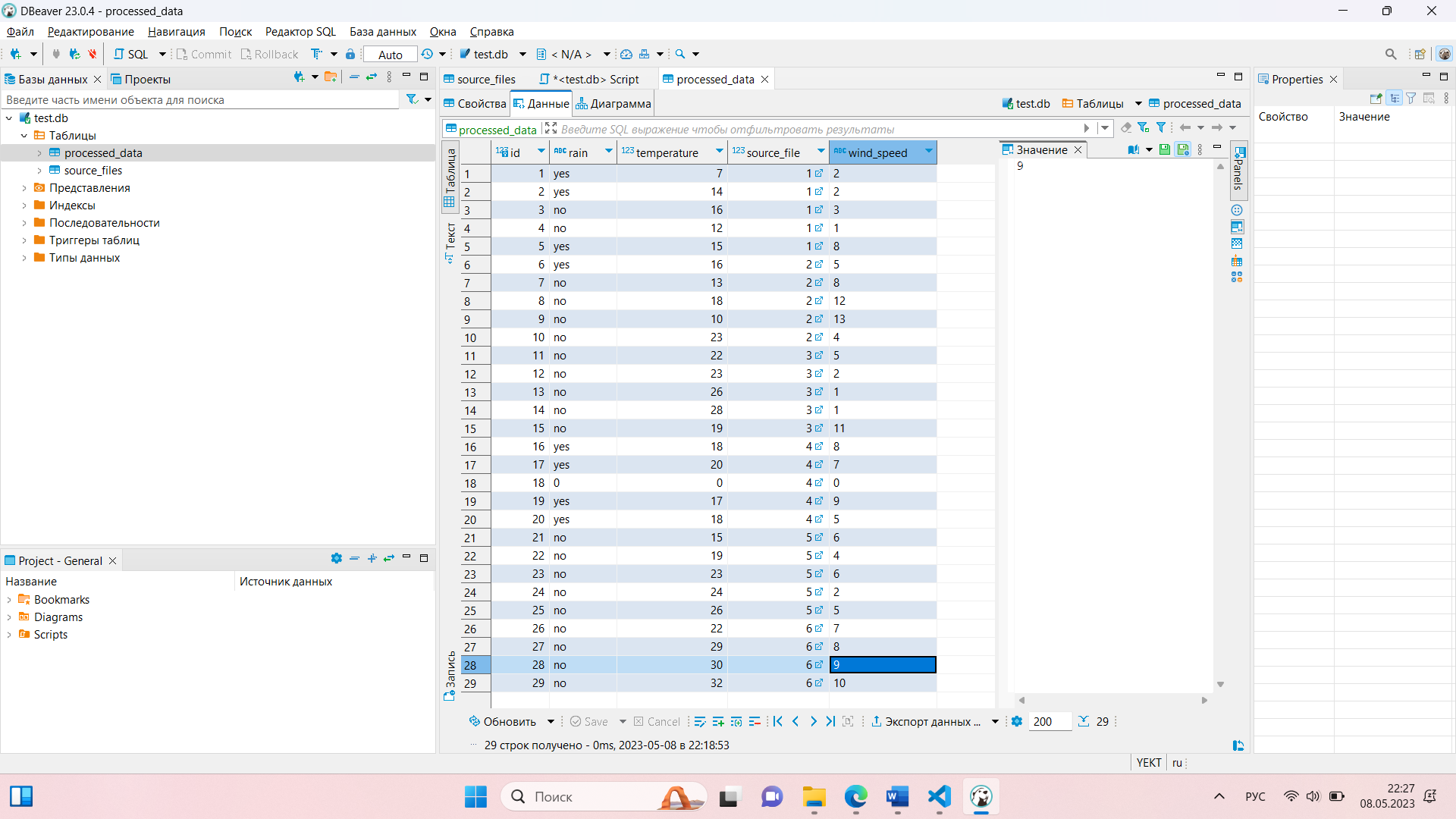
• Partial support for triggers. Available only FOR EACH ROW triggers.

• Unable to record in VIEWS. VIEWS they are read-only.

• Due to the implementation of the database as a single file and a departure from the concept of "client-server", access rights management capabilities are not supported (GRANT and REVOKE).

• By default, foreign keys are disabled.

Our database can be seemed on the screen below. There are 5 columns, in which there are information about weather. The whole request is possible thanks to sqlite3 library of python. I have added 4 CRUD functions in initial program.



**Web interface creation**

Based on the example, I created a website for my subject (weather forecast). Screenshot of the site is below:



The main tags and functionality of them:

• <main>the element is intended for contentunique to this page. The <main> block is only used one once on the page and placed inside <body>.

• <header> it represents a group of introductory content. If it is a child element of <body>, it defines the global title of the web page. It can include a logo and/or the name of a website or company. If <header If it is located as a child element of <article> or <section>, then it defines a specific title for that section (try not to confuse it with titles and headings).

• <nav> contains the main navigation functions for the page. Secondary links, etc. are not included in the navigation.

• <article> contains a block of linked content that contains the main semantic load of the page (for example, a table with data).

• <section>an auxiliary element that allows you to divide content in the <article> block into separate subsections. It usually contains a separate header as well. In these blocks, for example, you can separate a table with data and an additional description of the table, callouts, etc.

• <aside>this auxiliary element includes content that is not directly related to the main content but may contain additional information that is indirectly related to it: dictionary, author's biography, related links, and so on.

• <footer> it represents a group of final content for a page, the so-called "basement".

**Conclusion:** i have introduced to basic database design techniques, introduction to user interface design techniques.

Github: https://github.com/Ivan010403/-

Application

SQL-script

create table source\_files (

id integer PRIMARY KEY autoincrement,

filename varchar(255) NOT NULL,

processed datetime

);

create table processed\_data (

id integer PRIMARY KEY autoincrement,

rain varchar(255) NOT NULL,

temperature integer,

source\_file integer NOT NULL,

wind\_speed varchar(255) NOT NULL,

CONSTRAINT fk\_source\_files

FOREIGN KEY (source\_file)

REFERENCES source\_files(id)

ON DELETE CASCADE

);

CRUD functions

def get\_data\_from\_db(self, day):

try:

sqlite\_connection = sqlite3.connect('test.db')

cursor = sqlite\_connection.cursor()

print("Подключен к SQLite")

sqlite\_select\_query = """SELECT \* from processed\_data"""

cursor.execute(sqlite\_select\_query)

records = cursor.fetchall()

counter = 0

for row in records:

if counter == day-1:

print(row)

counter = counter + 1

except Exception as e:

print(e)

def put\_data\_into\_db(self, day):

sqlite\_connection = sqlite3.connect('test.db')

cursor = sqlite\_connection.cursor()

sql\_update\_query = """Update processed\_data set rain = ? where id = ?"""

data = (input("Enter rain "), day)

cursor.execute(sql\_update\_query, data)

sql\_update\_query = """Update processed\_data set temperature = ? where id = ?"""

data = (input("Enter temperature "), day)

cursor.execute(sql\_update\_query, data)

sql\_update\_query = """Update processed\_data set wind\_speed = ? where id = ?"""

data = (input("Enter wind\_speed "), day)

cursor.execute(sql\_update\_query, data)

sqlite\_connection.commit()

self.get\_data\_from\_db(day)

def new\_row(self):

sqlite\_connection = sqlite3.connect('test.db')

cursor = sqlite\_connection.cursor()

sqlite\_insert\_query = """INSERT INTO processed\_data

(id, rain, temperature, source\_file, wind\_speed)

VALUES

(30, 'no', 0, 0, 0);"""

count = cursor.execute(sqlite\_insert\_query)

sqlite\_connection.commit()

def delete\_row(self):

sqlite\_connection = sqlite3.connect('test.db')

cursor = sqlite\_connection.cursor()

sql\_delete\_query = """DELETE from processed\_data where id = 30"""

cursor.execute(sql\_delete\_query)

sqlite\_connection.commit()