БГУИР

Кафедра ЭВМ

Отчет по лабораторной работе № 5

Тема: «Работа с базой данных»

Выполнили:

студенты гр. №150502

Альхимович Н.Г.

Скалозуб К.А.

Проверила:

Герман Ю.О.

Минск

2023

**1 ЦЕЛЬ РАБОТЫ**

Изучить технику работы с базой данных в Scala.

**2 КРАТКИЕ ТЕОРЕТИЧЕСКИЕ СВЕДЕНИЯ**

Создаем базу данных командой:

create database mydb;

Подключаем ее командой:

use mydb;

Потом создаете простую таблицу warehouse с двумя столбцами: product и price:

create table warehouse(product varchar(15) primary key, price int(3));

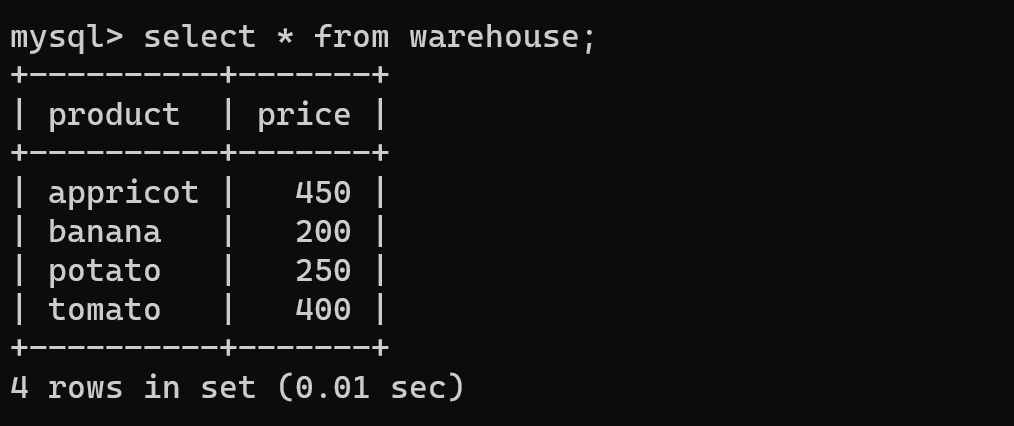
Заполняем ее данными:

insert into warehouse values("banana", 200);

insert into warehouse values("potato", 250);

insert into warehouse values("appricot", 450);

insert into warehouse values("tomato", 400);



Создаем проект в Eclipse Oxigen, который будет создавать окно приложения с двумя кнопками и полями для ввода, позволяющими вводить данные в базу данных и получать содержимое строк по ключу:

import java.awt.event.{ActionEvent, ActionListener}

import javax.swing.{JButton, JFrame, JPanel, SwingUtilities}

import java.awt.Dimension

import javax.swing.{JFrame, JLabel,JTextField, JPanel, SwingUtilities}

import java.sql.{Connection, DriverManager, ResultSet}

import java.awt.Color

object ButtonModule {

val frame = new JFrame("My Application")

frame.setSize(800, 600)

frame.setLayout(null)

def main(args: Array[String]): Unit = {

SwingUtilities.invokeLater(() => {

val label\_name = new JLabel("Title")

val label\_price = new JLabel("Price")

label\_name.setBounds(50,120,100,20)

label\_price.setBounds(200,120,70,20)

val insert = new JButton("Insert")

val select = new JButton("Select")

insert.setBounds(20,80,120,20)

select.setBounds(150,80,150,20)

val textField = new JTextField(20)

val textField2 = new JTextField(20)

textField.setBounds(20,150,130,20)

textField2.setBounds(160,150,130,20)

frame.add(label\_name)

frame.add(label\_price)

frame.add(insert)

frame.add(select)

frame.add(textField)

frame.add(textField2)

insert.addActionListener(new ActionListener {

override def actionPerformed(e: ActionEvent): Unit = {

val url = "jdbc:mysql://localhost:3306/mydb"

val username = "root"

val password = "nina"

Class.forName("com.mysql.jdbc.Driver")

val conn = DriverManager.getConnection(url, username, password)

try {

val stmt = conn.createStatement()

val prod\_name = textField.getText().trim()

val price = textField2.getText().trim()

val res\_check = stmt.executeQuery(s"SELECT product FROM warehouse WHERE product = '$prod\_name'")

if (res\_check.next()) {

textField.setText("Value already exists")

} else {

val res = stmt.executeUpdate("INSERT INTO warehouse VALUES ('$prod\_name', '$price')")

if (res > 0) {

textField2.setText("")

textField.setText("Added record")

} else {

textField.setText("Failed to add record")

}

}

} finally {

conn.close()

}

}

})

select.addActionListener(new ActionListener {

override def actionPerformed(e: ActionEvent): Unit = {

{

val url = "jdbc:mysql://localhost:3306/mydb"

val username = "root"

val password = "nina"

Class.forName("com.mysql.jdbc.Driver")

val conn = DriverManager.getConnection(url, username, password)

try {

val stmt = conn.createStatement()

val prod\_name= textField.getText().toString().trim()

val rs = stmt.executeQuery("SELECT \* FROM warehouse WHERE product = '"+prod\_name+"'")

while (rs.next()) {

val name = rs.getString("product")

val price = rs.getInt("price")

textField2.setText("" + price)

}

} finally {

conn.close()

}

}

}

})

frame.setBackground(Color.BLUE)

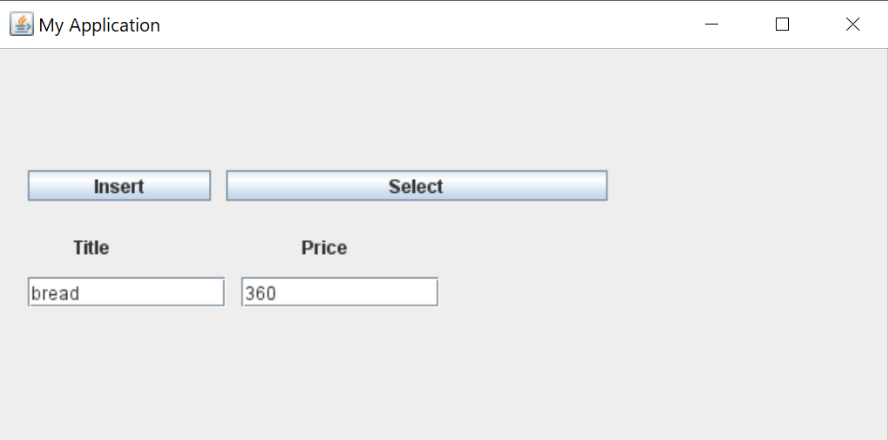
frame.setLocationRelativeTo(null)

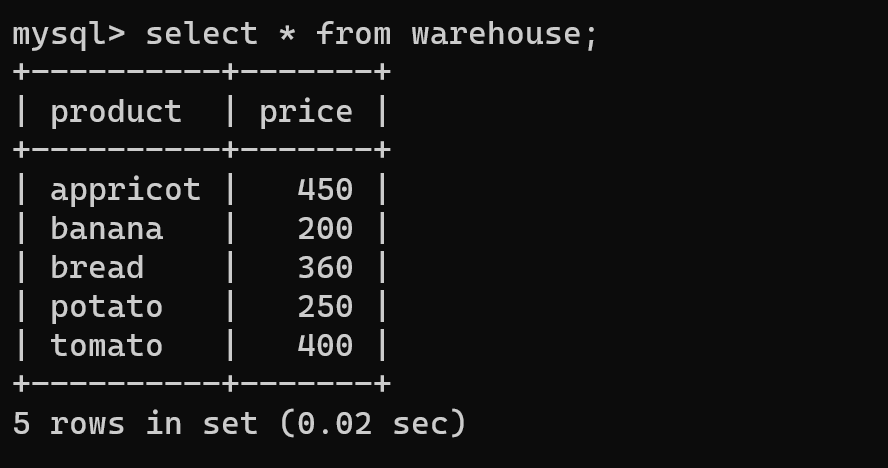
frame.setVisible(true)

})

}

}





1. **ХОД РАБОТЫ**
2. Расширить количество столбцов таблицы warehouse, добавив поле количество товара на складе.

Добавлен столбец quantity в базу данных:

ALTER TABLE warehouse

-> ADD COLUMN quantity int

-> ;

Измененная база данных представлена на рисунке 3.1:

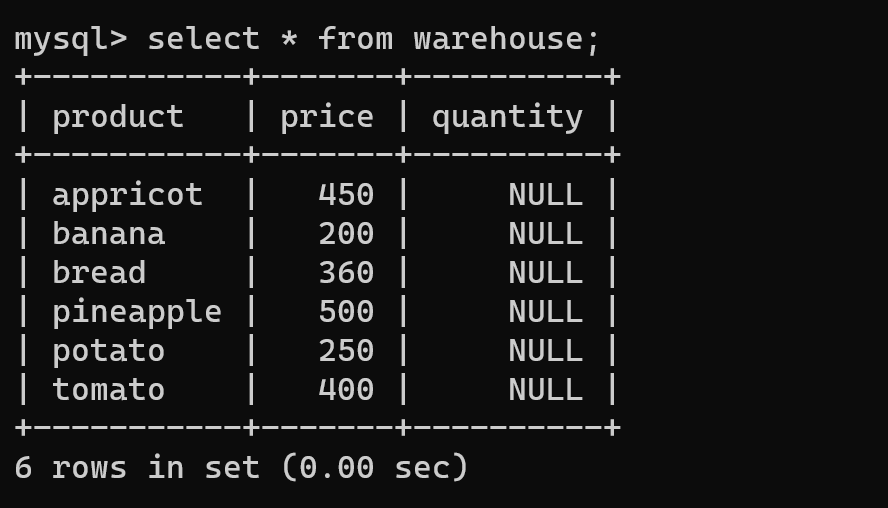


Рисунок 3.1 – База данных с добавленным столбцом

В код были внесены соответствующие изменения.

С помощью frame.add(label\_quant) и frame.add(textField3) добавлены поля:

val label\_quant = new JLabel("Quantity")

val textField3 = new JTextField(20)

Установлены их параметры:

label\_quant.setBounds(330,120,70,20)

textField3.setBounds(300,150,130,20)

Слушатель кнопки insert дополнен и изменена инициализация res:

val quantity = textField3.getText().trim()

…

val res = stmt.executeUpdate("INSERT INTO warehouse VALUES ('$prod\_name', '$price', '$quantity')")

…

textField3.setText("")

Слушатель кнопки select дополнен:

val quantity = rs.getInt("quantity")

textField3.setText("" + quantity)

Результат работы программы приведен на рисунках 3.2-3.5.

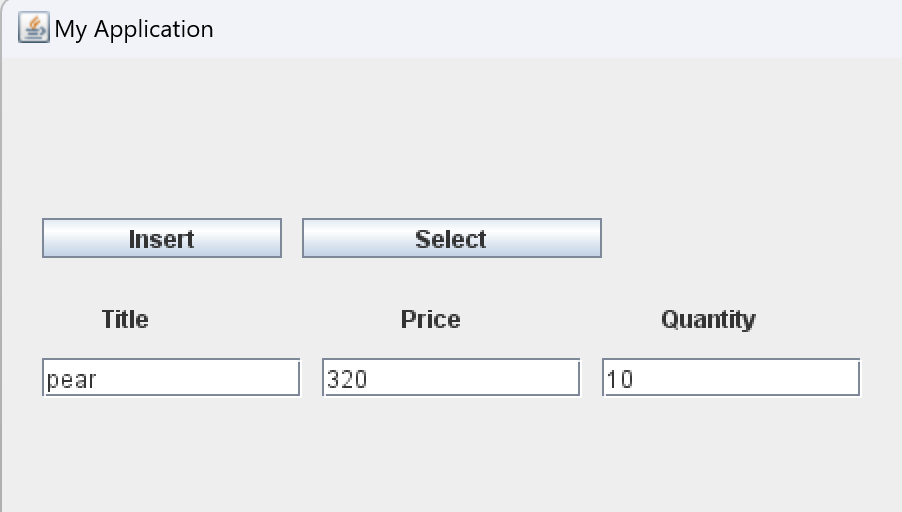


Рисунок 3.2 – Введены новые данные, нажата кнопка Insert

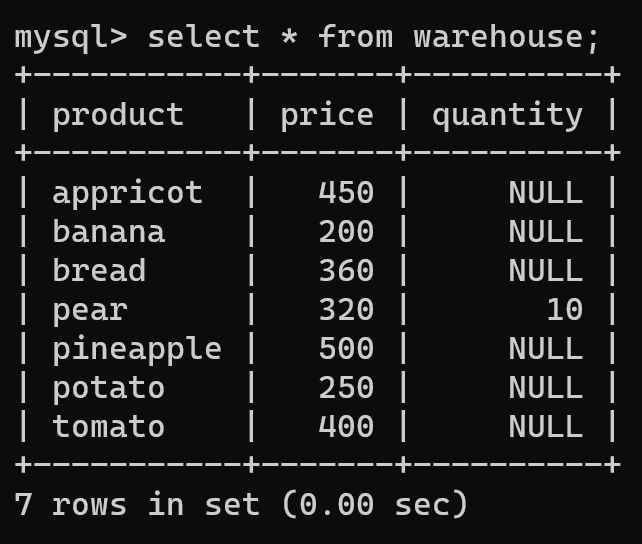


Рисунок 3.3 – Дополненная новой строкой база данных

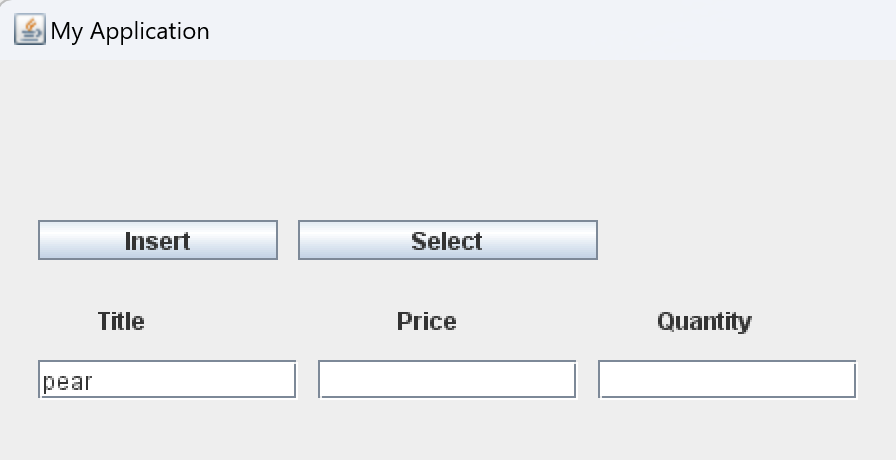


Рисунок 3.4 – Введен товар для получения данных, нажата кнопка Select

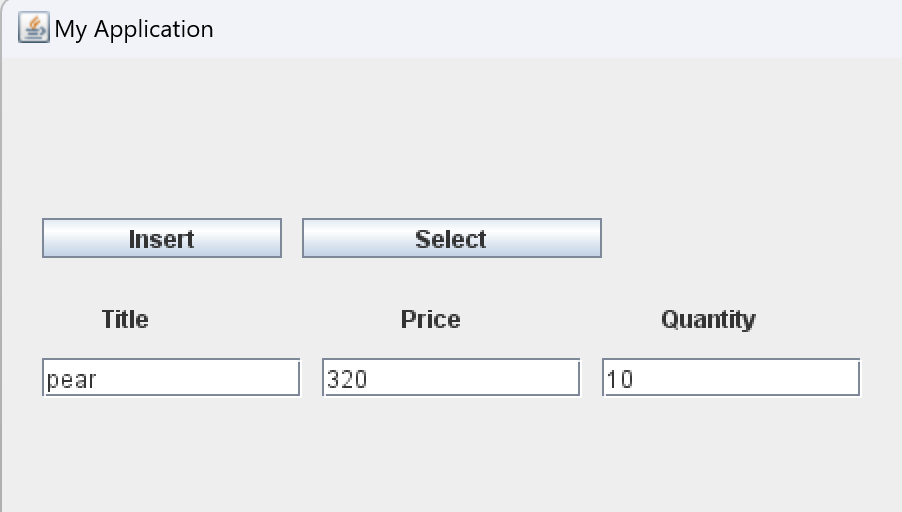


Рисунок 3.5 – Отображены данные о товаре из базы данных

1. Вывести товар, дающий максимальную прибыль (количество\*цена).

Изменения в коде программы описаны ниже.

Добавлена кнопка Max profit:

val get\_max = new JButton("Max profit")

get\_max.setBounds(310,80,150,20)

frame.add(get\_max)

Для нее реализован слушатель:

get\_max.addActionListener(new ActionListener {

override def actionPerformed(e: ActionEvent): Unit = {

{

val url = "jdbc:mysql://localhost:3306/mydb"

val username = "root"

val password = "nina"

Class.forName("com.mysql.jdbc.Driver")

val conn = DriverManager.getConnection(url, username, password)

try {

val stmt = conn.createStatement()

val rs = stmt.executeQuery("SELECT product FROM warehouse WHERE (price \* quantity) = (SELECT MAX(price \* quantity) FROM warehouse)")

if (rs.next()) {

val prod\_name = rs.getString("product")

textField2.setText("")

textField3.setText("")

textField.setText(prod\_name)

} else {

textField.setText("No data found")

}

} finally {

conn.close()

}

}

}

})

Результат работы программы приведен на рисунке 3.6.

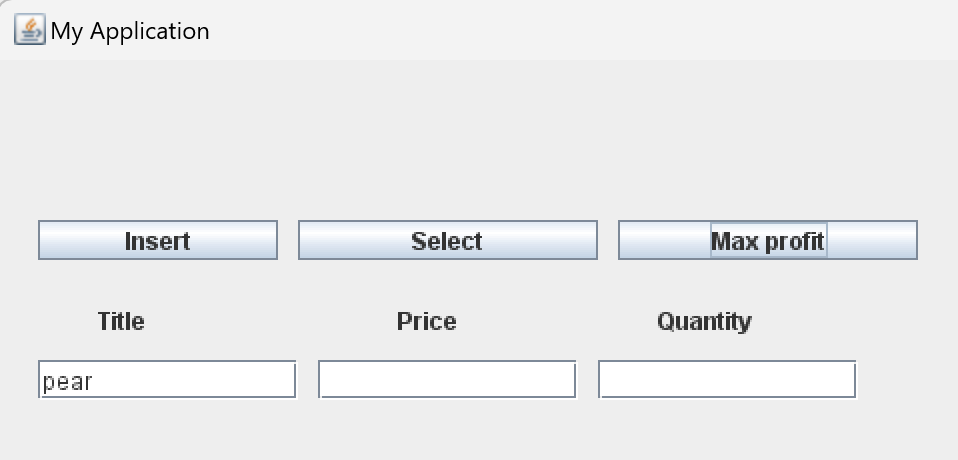


Рисунок 3.6 – Вывод наиболее прибыльного товара

1. Выбрать товар по ограничению (не меньше определенной величины).

Изменения в коде программы описаны ниже. Поиск товара реализован на основе анализа введенного параметра price, quantity или обоих.

Добавлена кнопка Find:

val get\_max = new JButton("Find")

get\_max.setBounds(470,80,150,20)

frame.add(find)

Для нее реализован слушатель:

find.addActionListener(new ActionListener {

override def actionPerformed(e: ActionEvent): Unit = {

{

val url = "jdbc:mysql://localhost:3306/mydb"

val username = "root"

val password = "nina"

Class.forName("com.mysql.jdbc.Driver")

val conn = DriverManager.getConnection(url, username, password)

try {

val stmt = conn.createStatement()

val price\_cond = textField2.getText().toString().trim()

val quant\_cond = textField3.getText().toString().trim()

val query =

if (price\_cond.nonEmpty && quant\_cond.nonEmpty) {

s"SELECT product FROM warehouse WHERE price >= $price\_cond AND quantity >= $quant\_cond"

} else if (price\_cond.nonEmpty) {

s"SELECT product FROM warehouse WHERE price >= $price\_cond"

} else if (quant\_cond.nonEmpty) {

s"SELECT product FROM warehouse WHERE quantity >= $quant\_cond"

} else {

"SELECT product FROM warehouse"

}

val rs = stmt.executeQuery(query)

if (rs.next()) {

val prod\_name = rs.getString("product")

textField.setText(prod\_name)

} else {

textField.setText("No data found")

}

} finally {

conn.close()

}

}

}

})

Результат работы программы приведен на рисунке 3.7.

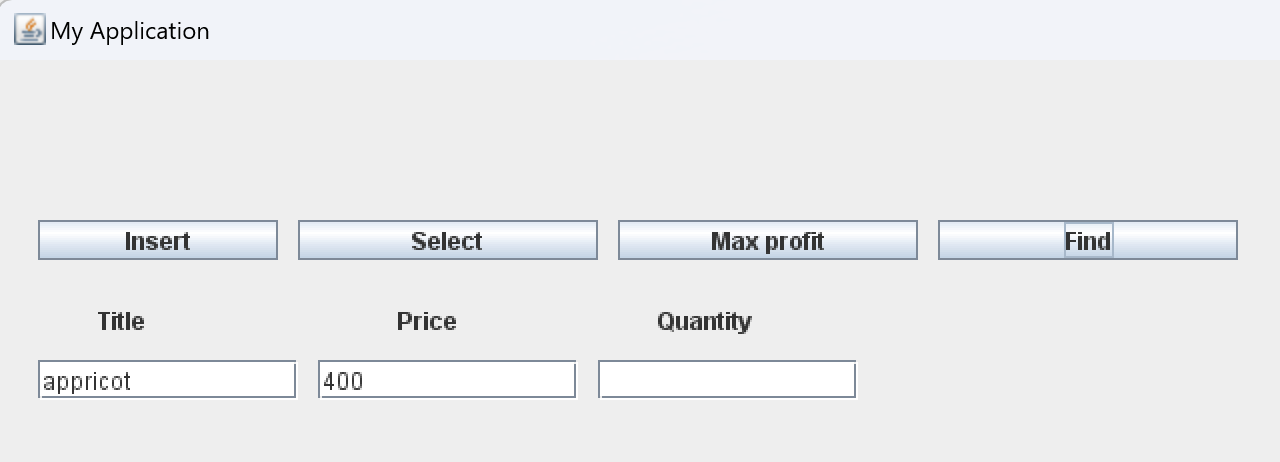


Рисунок 3.7 – Поиск товара стоимостью не менее 400

**4 ВЫВОДЫ**

В процессе выполнения лабораторной работы была изучена техника работы с базами данных в Scala и на основе полученных знаний написано приложение, осуществляющих обработку информации, хранящейся в базе данных.