МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ   
ФЕДЕРАЦИИ

ФГБОУ ВО «БАЙКАЛЬСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»

Кафедра математических методов и цифровых технологий

ОТЧЕТ

|  |  |  |
| --- | --- | --- |
| Студента бакалавриата группы | БИ-19-1 | Барлуков Н.В |

|  |  |  |
| --- | --- | --- |
| Руководитель(-и)  от университета |  | Родионов А.В |

Иркутск, 2021

**Лабораторная работа №4**

**Работа с данными**

**Вариант**

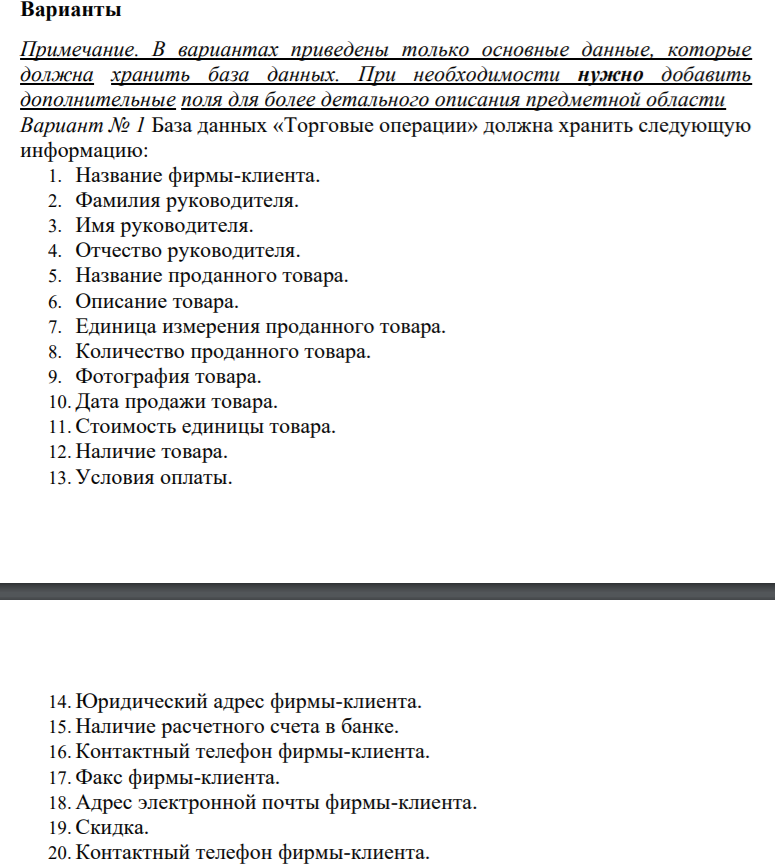


Схема базы данных:



Код

Класс Product

from sqlite3 import Connection, Cursor

class Product:

    def \_\_init\_\_(self, id, name, pic, measure,

        cost\_per\_unit, description, availability):

        self.id = id

        self.name = name

        self.pic = pic

        self.measure = measure

        self.cost\_per\_unit = cost\_per\_unit

        self.description = description

        self.availability = availability

    def get\_string(self):

        return f'''

        ИД: {self.id}

        Название: {self.name}

        Фото: {self.pic}

        Единица измерения: {self.measure}

        Цена за единицу: {self.cost\_per\_unit}

        Описание: {self.description}

        Наличие товара: {self.availability}

        '''

    def SetAvailability(self, conn: Connection, cursor: Cursor, availability):

        try:

            query = """

            UPDATE Products SET Availability = ?

            WHERE id = ?;

            """

            item\_tuple = (availability, self.id)

            cursor.execute(query, item\_tuple)

            conn.commit()

        except Exception as err:

            return err

Класс Products

import pandas as pd

from sqlite3 import Connection, Cursor

from models.Product import Product

class Products:

    def \_\_init\_\_(self, conn: Connection, cursor: Cursor):

        super().\_\_init\_\_()

        self.\_\_conn = conn

        self.\_\_cursor = cursor

    def get\_by\_id(self, id: int):

        self.\_\_cursor.execute("""

        SELECT ProductID, Name, Pic, Measure, CostPerUnit, Description, Availability

        FROM products

        WHERE ProductID = ?;

        """, [id])

        res = self.\_\_cursor.fetchone()

        try:

            return Product(res[0], res[1], res[2], res[3], res[4], res[5], res[6])

        except:

            return None

    def get\_all(self):

        self.\_\_cursor.execute("SELECT \* FROM Products;")

        rows = self.\_\_cursor.fetchall()

        id\_arr = []

        names\_arr = []

        pics\_arr = []

        measures\_arr = []

        CostPerUnits\_arr = []

        descriptions\_arr = []

        availability\_arr = []

        for row in rows:

            id\_arr.append(row[0])

            names\_arr.append(row[1])

            pics\_arr.append(row[2])

            measures\_arr.append(row[3])

            CostPerUnits\_arr.append(row[4])

            descriptions\_arr.append(row[5])

            availability\_arr.append(row[6])

        return pd.DataFrame(

            {"ИД": id\_arr,

            "Название": names\_arr,

            "Фото": pics\_arr,

            "Единица измерения": measures\_arr,

            "Цена за единицу": CostPerUnits\_arr,

            "Описание": descriptions\_arr,

            "Наличие": availability\_arr},

            index=None)

    def add(self, product: Product):

        query = """

        INSERT INTO products (Name, Pic, Measure, CostPerUnit, Description, Availability)

        VALUES (?, ?, ?, ?, ?, ?);

        """

        item\_tuple = (product.name, product.pic, product.measure, product.cost\_per\_unit,

            product.description, product.availability)

        self.\_\_cursor.execute(query, item\_tuple)

        self.\_\_conn.commit()

    def delete(self, id: int):

        try:

            query = """DELETE from Products where ProductID = ?;"""

            self.\_\_cursor.execute(query, [id])

            self.\_\_conn.commit()

            return None

        except Exception as err:

            return err

    def edit(self, id: int, product: Product):

        query = """

        UPDATE Products SET Name = ?, Pic = ?, Measure = ?,

        CostPerUnit = ?, Description = ?, Availability = ?

        WHERE ProductID = ?;

        """

        item\_tuple = (product.name, product.pic, product.measure, product.cost\_per\_unit,

            product.description, product.availability, id)

        self.\_\_cursor.execute(query, item\_tuple)

        self.\_\_conn.commit()

Класс Company

class Company:

    def \_\_init\_\_(self, id, companyName, firstName, lastName, createdAt, patronymic,address,phoneNumber,

    fax, email, checkingAccount ):

        self.id = id

        self.companyName = companyName

        self.firstName = firstName

        self.lastName = lastName

        self.createdAt = createdAt

        self.patronymic = patronymic

        self.address = address

        self.phoneNumber = phoneNumber

        self.fax = fax

        self.email = email

        self.checkingAccount = checkingAccount

    def get\_string(self):

        return f'''

        id: {self.id}

        Название фирмы-клиента: {self.companyName}

        Имя руководителя: {self.firstName}

        Фамилия руководителя: {self.lastName}

        Дата создания: {self.createdAt}

        Отчество руководителя: {self.patronymic}

        Юридический адрес фирмы: {self.address}

        Контактный телефон фирмы клиента: {self.phoneNumber}

        Факс фирмы клиента: {self.fax}

        Адрес электронной почты клиента: {self.email}

        Наличие расчетного счета в банке: {self.checkingAccount}

        '''

Класс Companies

from sqlite3 import Connection, Cursor

from models.Company import Company

import pandas as pd

class Companies:

    def \_\_init\_\_(self, conn: Connection, cursor: Cursor):

        super().\_\_init\_\_()

        self.\_\_conn = conn

        self.\_\_cursor = cursor

    def get\_by\_id(self, id: int):

        self.\_\_cursor.execute("""

        SELECT CompanyID , CompanyName, FirstName, LastName,

        created\_at, Patronymic, address, phoneNumber, Fax, Email, checkingAccount FROM Companies

        WHERE CompanyID = ?;

        """, [id])

        res = self.\_\_cursor.fetchone()

        try:

            return Company(res[0], res[1], res[2], res[3],

            res[4], res[5], res[6], res[7], res[8], res[9], res[10])

        except:

            return None

    def get\_all(self):

        self.\_\_cursor.execute("SELECT \* FROM Companies;")

        rows = self.\_\_cursor.fetchall()

        id\_arr = []

        first\_name\_arr = []

        name\_arr = []

        last\_name\_arr = []

        patronymic\_arr = []

        created\_at\_arr = []

        address\_arr = []

        phone\_number\_arr = []

        fax\_arr = []

        email\_arr = []

        checkingAccount\_arr = []

        for row in rows:

            id\_arr.append(row[0])

            name\_arr.append(row[1])

            first\_name\_arr.append(row[2])

            last\_name\_arr.append(row[3])

            created\_at\_arr.append(row[4])

            patronymic\_arr.append(row[5])

            address\_arr.append(row[6])

            phone\_number\_arr.append(row[7])

            fax\_arr.append(row[8])

            email\_arr.append(row[9])

            checkingAccount\_arr.append(row[10])

        return pd.DataFrame(

            {"ИД": id\_arr,

            "Название": name\_arr,

            "Имя руководителя": first\_name\_arr,

            "Фамилия руководителя": last\_name\_arr,

            "Отчество руководитель": patronymic\_arr,

            "Дата создания": created\_at\_arr,

            "Адрес": address\_arr,

            "Номер телефона": phone\_number\_arr,

            "Факс": fax\_arr,

            "Электронная почта": email\_arr,

            "Счет в банке": checkingAccount\_arr},

            index=None)

    def add(self, company: Company):

        query = """

        INSERT INTO Companies (CompanyName, FirstName, LastName,

        Patronymic, Address, PhoneNumber, Fax, Email, checkingAccount) VALUES (?, ?, ?, ?, ?, ?, ?, ?, ?);

        """

        item\_tuple = (company.companyName, company.firstName, company.lastName,

            company.patronymic, company.address, company.phoneNumber, company.fax, company.email,

            company.checkingAccount)

        self.\_\_cursor.execute(query, item\_tuple)

        self.\_\_conn.commit()

    def delete(self, id: int):

        try:

            query = "DELETE FROM Companies WHERE CompanyID = ?;"

            self.\_\_cursor.execute(query, [id])

            self.\_\_conn.commit()

            return None

        except Exception as err:

            return err

    def edit(self, id: int, company: Company):

        query = """

        UPDATE Companies SET CompanyName = ?, FirstName = ?, LastName = ?,

        Patronymic = ?, Address = ?, phoneNumber = ?, Fax = ?, Email = ?, checkingAccount = ?

        WHERE CompanyID = ?;

        """

        item\_tuple = (company.companyName, company.firstName, company.lastName,

            company.patronymic, company.address, company.phoneNumber, company.fax, company.email,

            company.checkingAccount, id)

        self.\_\_cursor.execute(query, item\_tuple)

        self.\_\_conn.commit()

Класс Sail

from sqlite3 import Connection, Cursor

from models.Company import Company

class Sail:

    def \_\_init\_\_(self, id ,companyID, productSailDate, paymentTerm,

    discount):

        self.id = id

        self.companyID = companyID

        self.productSailDate = productSailDate

        self.paymentTerm = paymentTerm

        self.discount = discount

    def get\_products(self, cursor: Cursor):

        cursor.execute("""

        SELECT p.name, pis.SailQuantity

        FROM Products as p JOIN ProductsInSails as pis ON p.ProductID = pis.ProductID

        WHERE pis.SailID = ?;

        """, [self.id])

        rows = cursor.fetchall()

        products = dict()

        for row in rows:

            products[row[0]] = row[1]

        return products

    def get\_company(self, cursor: Cursor):

        cursor.execute("""

        SELECT CompanyID, CompanyName, FirstName, LastName,

        Patronymic, created\_at, Address, phoneNumber, Fax, Email, checkingAccount FROM Companies

        WHERE CompanyID = ?;

        """, [self.companyID])

        res = cursor.fetchone()

        return Company(res[0], res[1], res[2], res[3],

            res[4], res[5], res[6], res[7], res[8], res[9], res[10])

    def get\_cost(self, cursor: Cursor):

        cursor.execute("""

        SELECT SUM(p.CostPerUnit \* pis.SailQuantity) - s.Discount as cost

        FROM Sails AS s

        JOIN ProductsInSails AS pis ON pis.SailID = s.SailID

        JOIN Products AS p ON pis.ProductID = p.ProductID

        WHERE pis.SailID = ?

        GROUP BY pis.SailID;""", [self.id])

        return cursor.fetchone()

    def get\_productSailDate(self, cursor : Cursor):

        cursor.execute("""

        SELECT ProductSailDate FROM Sails WHERE SailID = ?

        """, [self.id])

        return cursor.fetchone()

    def get\_products\_str(self, cursor: Cursor):

        return str(self.get\_products(cursor))

    def get\_company\_str(self, cursor: Cursor):

        company = self.get\_company(cursor)

        return f"{company.companyName} (ИД:{company.id})"

    def get\_string(self, cursor: Cursor):

        products = self.get\_products\_str(cursor)

        company = self.get\_company\_str(cursor)

        cost = self.get\_cost(cursor)

        return f'''

        ИД: {self.id}

        Товары: {products}

        Компания: {company}

        Cкидка: {self.discount}

        Дата: {self.productSailDate}

        Стоимость: {str(cost)}

        '''

    def get\_quantity(self, conn: Connection, cursor: Cursor, pr\_id: int):

        cursor.execute("""

        SELECT SailQuantity FROM ProductsInSails

        WHERE SailID = ? AND ProductID = ?;""", (self.id, pr\_id))

        return cursor.fetchone()

    def add\_product(self, conn: Connection, cursor: Cursor, pr\_id: int, quantity: int):

        query = """

        INSERT INTO ProductsInSails (SailID, ProductID, SailQuantity)

        VALUES (?, ?, ?);

        """

        cursor.execute(query, (self.id, pr\_id, quantity))

        conn.commit()

Класс Sails

import pandas as pd

from sqlite3 import Connection, Cursor

from models.Sail import Sail

class Sails:

    def \_\_init\_\_(self, conn: Connection, cursor: Cursor):

        super().\_\_init\_\_()

        self.\_\_conn = conn

        self.\_\_cursor = cursor

    def get\_by\_id(self, id: int):

        self.\_\_cursor.execute("""

        SELECT SailID, CompanyID, ProductSailDate, PaymentTerm, Discount FROM Sails

        WHERE SailID = ?;

        """, [id])

        res = self.\_\_cursor.fetchone()

        try:

            return Sail(res[0], res[1], res[2], res[3], res[4])

        except:

            return None

    def get\_all(self):

        self.\_\_cursor.execute("SELECT SailID FROM Sails;")

        rows = self.\_\_cursor.fetchall()

        id\_arr = []

        products\_arr = []

        companies\_arr = []

        productSailDate\_arr = []

        paymentTerm\_arr = []

        discount\_arr = []

        costs\_arr = []

        for row in rows:

            sail = self.get\_by\_id(row[0])

            id\_arr.append(sail.id)

            products\_arr.append(sail.get\_products\_str(self.\_\_cursor))

            companies\_arr.append(sail.get\_company\_str(self.\_\_cursor))

            productSailDate\_arr.append(sail.get\_productSailDate(self.\_\_cursor))

            paymentTerm\_arr.append(sail.paymentTerm)

            discount\_arr.append(sail.discount)

            costs\_arr.append(sail.get\_cost(self.\_\_cursor))

        return pd.DataFrame(

            {"ИД": id\_arr,

            "Товары": products\_arr,

            "Компания": companies\_arr,

            "Дата продажи": productSailDate\_arr,

            "Условия оплаты": paymentTerm\_arr,

            "Скидка": discount\_arr,

            "Стоимость": costs\_arr},

            index=None)

    def get\_next\_id(self):

        self.\_\_cursor.execute("""

        SELECT seq from sqlite\_sequence WHERE name = 'Sails';

        """)

        res = self.\_\_cursor.fetchone()

        return int(res[0]) + 1

    def add(self, sail: Sail):

        query = """

        INSERT INTO Sails (CompanyID, PaymentTerm, Discount)

        VALUES (?, ?, ?);

        """

        item\_tuple = (sail.companyID, sail.paymentTerm, sail.discount)

        self.\_\_cursor.execute(query, item\_tuple)

        self.\_\_conn.commit()

    def delete(self, id: int):

        try:

            query = """DELETE from Sails where SailID = ?;"""

            self.\_\_cursor.execute(query, [id])

            self.\_\_conn.commit()

            return None

        except Exception as err:

            return err

    def edit(self, id: int, sail : Sail):

        query = """

        UPDATE Sails SET CompanyID = ?, PaymentTerm = ?, Discount = ?

        WHERE SailID = ?;

        """

        item\_tuple = (sail.companyID, sail.paymentTerm, sail.discount, id)

        self.\_\_cursor.execute(query, item\_tuple)

        self.\_\_conn.commit()

Main

import sqlite3

from models.Companies import Companies

from models.Company import Company

from models.Products import Products

from models.Product import Product

from models.Sails import Sails

from models.Sail import Sail

main\_commands = {

    "q": -1,

    "help": 0,

    "companies" : 1,

    "products" : 2,

    "sails" : 3,

}

def help():

    print(

"""

- Чтобы перейти в управление компаниями введите "companies"

- Чтобы перейти в управление товарами введите "products"

- Чтобы перейти в управление продажами введите "sails"

- Чтобы посмотреть инструкцию, введите "help"

- Чтобы выйти, введите "q"

""")

cur = None

conn = None

companies = Companies(None, None)

sails = Sails(None, None)

products = Products(None, None)

def init\_connection():

    global conn, cur

    conn = sqlite3.connect("Database/Sailsoperations.db")

    cur = conn.cursor()

    global companies, sails, products

    companies = Companies(conn, cur)

    sails = Sails(conn, cur)

    products = Products(conn, cur)

def main\_menu():

    init\_connection()

    command = input("Введите команду: ")

    if command in main\_commands:

        i = main\_commands[command]

        if i == -1:

            exit()

        elif i == 0:

            help()

            main\_menu()

        elif i == 1:

            companies\_menu()

        elif i == 2:

            products\_menu()

        elif i == 3:

            sails\_menu()

    else:

        print('''\nНекорректный ввод...

        Прочитайте инструкцию по использованию данной программы:''')

        help()

        main\_menu()

def companies\_menu():

    try:

        global clients

        answer = int(input("Для того чтобы просмотреть список всех компаний, введите 1. Чтобы добавить компанию, введите 2. Чтобы удалить компанию, введите 3. Чтобы редактировать компанию, введите 4. (Чтобы вернуться назад нажмите 0)\n"))

        if answer == -1:

            exit()

        if answer == 0:

            main\_menu()

        if answer == 1:

            get\_companies()

        if answer == 2:

            add\_company()

        if answer == 3:

            del\_company()

        if answer == 4:

            edit\_company()

    except:

        print('Что-то пошло не так!')

        main\_menu()

def get\_companies():

    global companies

    print()

    print(companies.get\_all())

    companies\_menu()

def get\_company():

    global companies

    try:

        id = int(input('Введите ID компании: '))

        company = companies.get\_by\_id(id)

        if company:

            print(company.get\_string())

            companies\_menu()

        else:

            print('\nНет компании с заданным ИД.')

            companies\_menu()

    except:

        print('\nЧто-то пошло не так!')

        companies\_menu()

def add\_company():

    companyName = input('Введите название компании: ')

    firstName = input('Введите имя руководителя: ')

    lastName = input('Введите фамилию руководителя: ')

    patronymic = input('Введите отчество руководителя: ')

    address = input('Введите адрес компании: ')

    phone = input('Введите номер телефона компании: ')

    fax = input('Введите факс компании: ')

    email = input('Введите эл.почту компании: ')

    checkingAccount = input('Введите наличие расчетного счета в банке: ')

    global companies

    company = Company(0, companyName, firstName, lastName, None, patronymic,

        address, phone, fax, email, checkingAccount)

    companies.add(company)

    print('Компания успешно создана.')

    companies\_menu()

def del\_company():

    try:

        global companies

        id = int(input('Введите ID компании: '))

        company = companies.get\_by\_id(id)

        if not company:

            print('Нет компании с таким ИД.')

            companies\_menu()

        ans = input(f'Вы точно хотите удалить компанию №{id}? (y/n)\n')

        if ans == 'y':

            err = companies.delete(id)

            if err is None:

                print('Компания успешно удалена!')

                companies\_menu()

            else:

                print('Произошла ошибка при удалении компании: ' + str(err))

                companies\_menu()

        else:

            companies\_menu()

    except:

        print('Что-то пошло не так!')

        companies\_menu()

def edit\_company():

    try:

        global companies

        id = int(input('Введите ID компании: '))

        company = companies.get\_by\_id(id)

        if not company:

            print('Нет компании с таким ИД.')

            companies\_menu()

        companyName = input('Введите название компании: ')

        firstName = input('Введите имя руководителя: ')

        lastName = input('Введите фамилию руководителя: ')

        patronymic = input('Введите отчество руководителя: ')

        address = input('Введите адрес компании: ')

        phone = input('Введите номер телефона компании: ')

        fax = input('Введите факс компании: ')

        email = input('Введите эл.почту компании: ')

        checkingAccount = input('Введите наличие расчетного счета в банке: ')

        company = Company(0, companyName, firstName, lastName, None, patronymic,

            address, phone, fax, email, checkingAccount)

        companies.edit(id, company)

        print('Компания успешно изменена.')

        companies\_menu()

    except:

        print('Что-то пошло не так!')

        companies\_menu()

def products\_menu():

    try:

        global clients

        answer = int(input("Для того чтобы просмотреть список всех товаров, введите 1. Чтобы добавить товар, введите 2. Чтобы удалить товар, введите 3. Чтобы редактировать товар, введите 4. (Чтобы вернуться назад нажмите 0)\n"))

        if answer == -1:

            exit()

        if answer == 0:

            main\_menu()

        if answer == 1:

            get\_products()

        if answer == 2:

            add\_product()

        if answer == 3:

            del\_product()

        if answer == 4:

            edit\_product()

    except:

        print('Что-то пошло не так!')

        main\_menu()

def get\_products():

    global products

    print()

    print(products.get\_all())

    products\_menu()

def get\_product():

    global products

    try:

        id = int(input('Введите ID товара: '))

        product = products.get\_by\_id(id)

        if product:

            print(product.get\_string())

            products\_menu()

        else:

            print('\nНет товара с заданным ИД.')

            products\_menu()

    except:

        print('\nЧто-то пошло не так!')

        products\_menu()

def add\_product():

    try:

        name = input('Введите название товара: ')

        pic = input('Введите ссылку на фото: ')

        measure = input('Введите единицу изерения: ')

        cost\_per\_unit = float(input('Введите цену за единицу: '))

        description = input('Введите описание: ')

        availability = input('Введите наличие товара: ')

    except:

        print('\nВы ввели данные неправильного формата')

        products\_menu()

    global products

    product = Product(0, name, pic, measure, cost\_per\_unit, description, availability)

    products.add(product)

    print('Товар успешно создан.')

    products\_menu()

def del\_product():

    try:

        global products

        id = int(input('Введите ID товара: '))

        product = products.get\_by\_id(id)

        if not product:

            print('Нет товара с таким ИД.')

            products\_menu()

        ans = input(f'Вы точно хотите удалить компанию №{id}? (y/n)\n')

        if ans == 'y':

            err = products.delete(id)

            if err is None:

                print('Товар успешно удален.')

                products\_menu()

            else:

                print('Произошла ошибка при удалении товара: ' + str(err))

                products\_menu()

        else:

            products\_menu()

    except:

        print('Что-то пошло не так!')

        products\_menu()

def edit\_product():

    try:

        global products

        id = int(input('Введите ID товара: '))

        product = products.get\_by\_id(id)

        if not product:

            print('Нет товара с таким ИД.')

            products\_menu()

        name = input('Введите название товара: ')

        pic = input('Введите ссылку на фото: ')

        measure = input('Введите единицу изерения: ')

        cost\_per\_unit = float(input('Введите цену за единицу: '))

        description = input('Введите описание: ')

        availability = input('Введите наличие товара: ')

        product = Product(0, name, pic, measure, cost\_per\_unit, description, availability)

        products.edit(id, product)

        print('Товар успешно изменен.')

        products\_menu()

    except:

        print('\nЧто-то пошло не так!')

        products\_menu()

def change\_availability():

    global products, conn, cur

    try:

        id = int(input('Введите ID товара: '))

        product = products.get\_by\_id(id)

        if not product:

            print('Нет товара с таким ИД.')

            products\_menu()

        availability = input('Введите наличие товара: ')

        err = product.SetAvailability(conn, cur, availability)

        if err is None:

            print('Наличие успешно изменено.')

            products\_menu()

        else:

            print('Произошла ошибка при изменении информации о товаре!')

            products\_menu()

    except ValueError:

        print('Вы ввели данные неправильного формата')

        products\_menu()

def sails\_menu():

    try:

        global clients

        answer = int(input("Для того чтобы просмотреть список всех продаж, введите 1. Чтобы добавить продажу, введите 2. Чтобы удалить продажу, введите 3. Чтобы редактировать продажу, введите 4. (Чтобы вернуться назад нажмите 0)\n"))

        if answer == -1:

            exit()

        if answer == 0:

            main\_menu()

        if answer == 1:

            get\_sails()

        if answer == 2:

            add\_sail()

        if answer == 3:

            del\_sail()

        if answer == 4:

            edit\_sail()

    except:

        print('Что-то пошло не так!')

        sails\_menu()

def get\_sails():

    global sails

    print()

    print(sails.get\_all())

    sails\_menu()

def get\_sail():

    global sails

    try:

        id = int(input('Введите ID продажи: '))

        operation = sails.get\_by\_id(id)

        if operation:

            print(operation.get\_string(cur))

            sails\_menu()

        else:

            print('\nНет продажи с заданным ИД.')

            sails\_menu()

    except:

        print('\nЧто-то пошло не так!')

        sails\_menu()

def add\_sail():

    try:

        global companies, products

        print(companies.get\_all())

        company\_id = int(input('Введите ИД компании: '))

        paymentTerm = input('Введите условия оплаты: ')

        n\_prods = input('Введите количество видов товаров: ')

        n\_prods = int(n\_prods)

        products\_dict = {}

        print(products.get\_all())

        for i in range(n\_prods):

            pr\_id = int(input('Введите ИД товара №' + str(i + 1) + ' в данной продаже: '))

            q = int(input('Введите количество единиц товара с ИД=' + str(pr\_id) + ': '))

            products\_dict[pr\_id] = q

        discount = float(input('Введите скидку: '))

    except:

        print('\nВы ввели данные неправильного формата')

        sails\_menu()

    global sails, conn, cur

    sail = Sail(sails.get\_next\_id(), company\_id, None, paymentTerm, discount)

    sails.add(sail)

    for key in products\_dict:

        err = sail.add\_product(conn, cur, key, products\_dict[key])

        if err is not None:

            break

    if err is None:

        print('Продажа успешно совершена.')

        sails\_menu()

    else:

        print('Произошла ошибка при совершении продажи: ' + str(err))

        sails\_menu()

def del\_sail():

    try:

        global sails

        id = int(input('Введите ID продажи: '))

        sail = sails.get\_by\_id(id)

        if not sail:

            print('Нет продажи с таким ИД.')

            sails\_menu()

        ans = input(f'Вы точно хотите удалить продажу №{id}? (y/n)\n')

        if ans == 'y':

            err = sails.delete(id)

            if err is None:

                print('Продажа успешно удалена.')

                sails\_menu()

            else:

                print('Произошла ошибка при удалении продажи: ' + str(err))

                sails\_menu()

        else:

            sails\_menu()

    except:

        print('\nЧто-то пошло не так!')

        sails\_menu()

def edit\_sail():

    try:

        global sails, conn, cur

        id = int(input('Введите ИД продажи: '))

        sail = sails.get\_by\_id(id)

        if not sail:

            print('Нет продажи с таким ИД.')

            sails\_menu()

        global companies, products

        print(companies.get\_all())

        company\_id = int(input('Введите ИД компании: '))

        paymentTerm = input('Введите условия оплаты: ')

        discount = input('Введите скидку: ')

        sail = Sail(id, company\_id, None, paymentTerm, discount)

        sails.edit(id, sail)

        print('Продажа успешно изменена.')

        sails\_menu()

    except:

        print('\nЧто-то пошло не так!')

        sails\_menu()

main\_menu()

Пример выполнения