Студент группы ИС-22 Дьяченко Л.А.

Практическое занятие № 15

Тема: составление программ в IDE PyCharm Community.

Цель: закрепить усвоенные знания, понятия, алгоритмы,

основные принципы составления программ, работы с БД в IDE PyCharm Community.

Постановка задачи:

Приложение ЗАКАЗЫ ТОВАРОВ для автоматизированного контроля заказов

торговой фирмы. Таблица Заказы должна содержать информацию о товарах со

следующей структурой записи: Код товара, Наименование товара, Заказчик

(наименование организации, возможны повторяющиеся значения), Дата заказа, Срок

исполнения (от 1 до 10 дней), Стоимость заказа.

Текст программы:

"""

Приложение ЗАКАЗЫ ТОВАРОВ для автоматизированного контроля заказов

торговой фирмы. Таблица Заказы должна содержать информацию о товарах со

следующей структурой записи: Код товара, Наименование товара, Заказчик

(наименование организации, возможны повторяющиеся значения), Дата заказа, Срок

исполнения (от 1 до 10 дней), Стоимость заказа.

"""

import sqlite3

from datetime import date, timedelta

# Подключение к базе данных

conn = sqlite3.connect('orders.db')

c = conn.cursor()

# Создание таблицы Заказы

c.execute('''CREATE TABLE IF NOT EXISTS Orders

             (OrderID INTEGER PRIMARY KEY AUTOINCREMENT, ProductCode TEXT, ProductName TEXT, Customer TEXT, OrderDate TEXT, DeliveryDate TEXT, OrderCost REAL)''')

# Функция для добавления нового заказа

def add\_order(product\_code, product\_name, customer, order\_date, delivery\_days, order\_cost):

    order\_date = date.fromisoformat(order\_date)

    delivery\_date = order\_date + timedelta(days=delivery\_days)

    c.execute("INSERT INTO Orders (ProductCode, ProductName, Customer, OrderDate, DeliveryDate, OrderCost) VALUES (?, ?, ?, ?, ?, ?)",

              (product\_code, product\_name, customer, order\_date.isoformat(), delivery\_date.isoformat(), order\_cost))

    conn.commit()

# Функция для просмотра всех заказов

def view\_orders():

    c.execute("SELECT \* FROM Orders")

    rows = c.fetchall()

    for row in rows:

        print(row)

# Функция для поиска заказов по критериям

def search\_orders(product\_code=None, customer=None, order\_date=None):

    query = "SELECT \* FROM Orders WHERE "

    conditions = []

    if product\_code:

        conditions.append(f"ProductCode = '{product\_code}'")

    if customer:

        conditions.append(f"Customer = '{customer}'")

    if order\_date:

        conditions.append(f"OrderDate = '{order\_date}'")

    if conditions:

        query += " AND ".join(conditions)

    else:

        query = "SELECT \* FROM Orders"

    c.execute(query)

    rows = c.fetchall()

    for row in rows:

        print(row)

# Функция для обновления заказа

def update\_order(order\_id, product\_code=None, product\_name=None, customer=None, order\_date=None, delivery\_days=None, order\_cost=None):

    updates = []

    if product\_code:

        updates.append(f"ProductCode = '{product\_code}'")

    if product\_name:

        updates.append(f"ProductName = '{product\_name}'")

    if customer:

        updates.append(f"Customer = '{customer}'")

    if order\_date:

        order\_date = date.fromisoformat(order\_date)

        updates.append(f"OrderDate = '{order\_date.isoformat()}'")

    if delivery\_days:

        if not order\_date:

            c.execute("SELECT OrderDate FROM Orders WHERE OrderID = ?", (order\_id,))

            order\_date\_str = c.fetchone()[0]

            order\_date = date.fromisoformat(order\_date\_str)

        delivery\_date = order\_date + timedelta(days=delivery\_days)

        updates.append(f"DeliveryDate = '{delivery\_date.isoformat()}'")

    if order\_cost:

        updates.append(f"OrderCost = {order\_cost}")

    update\_query = "UPDATE Orders SET " + ", ".join(updates) + f" WHERE OrderID = {order\_id}"

    c.execute(update\_query)

    conn.commit()

# Функция для удаления заказа

def delete\_order(order\_id):

    c.execute("DELETE FROM Orders WHERE OrderID = ?", (order\_id,))

    conn.commit()

# Примеры использования

add\_order('PR001', 'Product 1', 'Company A', '2023-03-27', 5, 100.0)

add\_order('PR002', 'Product 2', 'Company B', '2023-03-28', 3, 200.0)

add\_order('PR001', 'Product 1', 'Company A', '2023-03-29', 7, 150.0)

print("Все заказы:")

view\_orders()

print("\nОбновление заказа с ID 1:")

update\_order(1, product\_name='Updated Product 1', order\_cost=110.0)

view\_orders()

print("\nУдаление заказа с ID 2:")

delete\_order(2)

view\_orders()

# Закрытие соединения с базой данных

conn.close()

Протокол работы программы:

Все заказы:

(1, 'PR001', 'Product 1', 'Company A', '2023-03-27', '2023-04-01', 100.0)

(2, 'PR002', 'Product 2', 'Company B', '2023-03-28', '2023-03-31', 200.0)

(3, 'PR001', 'Product 1', 'Company A', '2023-03-29', '2023-04-05', 150.0)

Обновление заказа с ID 1:

(1, 'PR001', 'Updated Product 1', 'Company A', '2023-03-27', '2023-04-01', 110.0)

(2, 'PR002', 'Product 2', 'Company B', '2023-03-28', '2023-03-31', 200.0)

(3, 'PR001', 'Product 1', 'Company A', '2023-03-29', '2023-04-05', 150.0)

Удаление заказа с ID 2:

(1, 'PR001', 'Updated Product 1', 'Company A', '2023-03-27', '2023-04-01', 110.0)

(3, 'PR001', 'Product 1', 'Company A', '2023-03-29', '2023-04-05', 150.0)

Вывод: Я закрепил усвоенные знания, понятия, алгоритмы, основные принципы составления программ, работы с БД в IDE PyCharm Community.

PyCharm Community.