Министерство науки и высшего образования Российской Федерации
Федеральное государственное автономное образовательное учреждение высшего образования
«СЕВЕРО-КАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Институт перспективной инженерии
Департамент цифровых, робототехнических систем и электроники

ОТЧЕТ
ПО ЛАБОРАТОРНОЙ РАБОТЕ №11
дисциплины
«Объектно-ориентированное программирование»
Вариант № 18

Выполнил:
Текеева Мадина Азрет-Алиевна
3 курс, группа ИВТ-б-о-23-2,
09.03.01 «Информатика и вычислительная техника»,
направленность (профиль) «Программное обеспечение средств вычислительной техники и автоматизированных систем», очная форма обучения

_____
(подпись)

Проверил:
Доцент департамента цифровых, робототехнических систем и электроники института перспективной инженерии Воронкин Р.А

_____
(подпись)

Отчет защищен с оценкой _____ Дата защиты_____

Ставрополь, 2025 г.

Тема: Основы работы с SQLite3.

Цель работы: исследовать базовые возможности системы управления базами данных SQLite3.

Репозиторий: https://github.com/bickrosss/TM_oop_lab11

Порядок выполнения работы:

1. Создана база данных game_sales.db для работы с датасетом видеоигр.

```
@bickrosss →/workspaces/TM_oop_lab11 (main) $ sqlite3 game_sales.db
SQLite version 3.45.3 2024-04-15 13:34:05
Enter ".help" for usage hints.
sqlite>
```

Рисунок 1. Подключение к новой базе данных

2. Создали таблицу «Customer», показали ее содержимое, показали список и структуру всех таблиц в базе.

```
sqlite> CREATE TABLE customer(name varchar(50));
sqlite> SELECT * FROM customer;
sqlite> .schema customer
CREATE TABLE customer(name varchar(50));
```

Рисунок 2. Работа с таблицей «Customer»

3. Нашли команду, которая отвечает за вывод времени выполнения запроса.

```
sqlite> .timer on
sqlite> SELECT COUNT(*) FROM city;
55
Run Time: real 0.000 user 0.000090 sys 0.000000
```

Рисунок 3. Вывод времени

4. Загрузили csv файл в созданную базу данных. Вывод максимальной длины названия города.

```
sqlite> .import -- csv city.csv city
ERROR: unknown option: "-".  Usage:
.import FILE TABLE       Import data from FILE into TABLE
   Options:
     --ascii               Use \037 and \036 as column and row separators
     --csv                 Use , and \n as column and row separators
     --skip N              Skip the first N rows of input
     --schema S            Target table to be S.TABLE
     -v                    "Verbose" - increase auxiliary output
   Notes:
     *  If TABLE does not exist, it is created.  The first row of input
        determines the column names.
     *  If neither --csv or --ascii are used, the input mode is derived
        from the ".mode" output mode
     *  If FILE begins with "|" then it is a command that generates the
        input text.
sqlite> SELECT MAX(LENGTH(city)) FROM city;
16
```

Рисунок 4. Загрузка csv файла

5. Загрузили csv файл в созданную базу данных без опции --csv.

```
sqlite> .mode csv
sqlite> .import datasets/city.csv city
sqlite> SELECT MAX(LENGTH(city)) FROM city;
16
```

Рисунок 5. Загрузка csv файла без опции --csv

6. Вывели данные о количестве городов для каждого часового пояса в Сибирском и Приволжском федеральных округах.

```
sqlite> SELECT timezone, COUNT(*) AS city_count FROM city
   ...> WHERE federal_district IN ('Сибирский', 'Приволжский')
   ...> GROUP BY timezone ORDER BY timezone;
+03:00,15
+04:00,9
+05:00,9
+06:00,9
+07:00,6
+08:00,3
```

Рисунок 6. Вывод данных о количестве городов

7. Сформировали пять запросов к таблицам БД.

Запрос 1: Топ-10 игр по мировым продажам.

```sql
SELECT
    Name,
    Platform,
    Year,
    ROUND(Global_Sales, 2) AS Sales_M
FROM video_ games
ORDER BY Global_Sales DESC
LIMIT 10;
```

1	Name	Platform	Year	Global_Sales
2	Wii Sports	Wii	2006.0	82.74
3	Super Mario Bros.	NES	1985.0	40.24
4	Mario Kart Wii	Wii	2008.0	35.82
5	Wii Sports Resort	Wii	2009.0	33.0
6	Pokemon Red/Pokemon Blue	GB	1996.0	31.37
7	Tetris	GB	1989.0	30.26
8	New Super Mario Bros.	DS	2006.0	30.01
9	Wii Play	Wii	2006.0	29.02
10	New Super Mario Bros. Wii	Wii	2009.0	28.62
11	Duck Hunt	NES	1984.0	28.31

Рисунок 7. Результат первого запроса

Запрос 2: Распределение игр по жанрам.

```sql
SELECT
    Genre,
    COUNT(*) AS Games_Count,
    SUM(Global_Sales) AS Total_Sales,
    AVG(Global_Sales) AS Avg_Sales
FROM video_games
GROUP BY Genre
ORDER BY Total_Sales DESC;
```

1	Genre	Games_Count	Total_Sales	Avg_Sales
2	Action	106	563.0	5.311320754716981
3	Shooter	76	487.12	6.409473684210527
4	Platform	57	433.8	7.610526315789474
5	Role-Playing	58	398.94	6.878275862068966
6	Sports	59	395.71999999999997	6.707118644067796
7	Misc	43	268.49	6.243953488372093
8	Racing	34	256.26	7.537058823529412
9	Simulation	19	111.65	5.876315789473685
10	Fighting	24	111.56	4.648333333333333
11	Puzzle	11	87.62	7.965454545454546
12	Adventure	8	38.79	4.84875
13	Strategy	5	23.2	4.64

Рисунок 8. Результат второго запроса

Запрос 3: Игры Nintendo после 2010 года.

```
SELECT
    Name,
    Year,
    Platform,
    Global_Sales
FROM video_games
WHERE Publisher = 'Nintendo'
    AND Year >= 2010
    AND Year IS NOT NULL
ORDER BY Year DESC, Global_Sales DESC;
```

1	Name	Year	Platform	Global_Sales
2	Splatoon	2015.0	WiiU	4.57
3	Super Mario Maker	2015.0	WiiU	3.18
4	Animal Crossing: Happy Home Designer	2015.0	3DS	2.98
5	Pokemon Omega Ruby/Pokemon Alpha Sapphire	2014.0	3DS	11.33
6	Super Smash Bros. for Wii U and 3DS	2014.0	3DS	7.45
7	Mario Kart 8	2014.0	WiiU	6.96
8	Super Smash Bros. for Wii U and 3DS	2014.0	WiiU	5.02
9	Monster Hunter 4 Ultimate	2014.0	3DS	3.89
10	Yokai Watch 2 Ganso/Honke	2014.0	3DS	3.22
11	Pokemon X/Pokemon Y	2013.0	3DS	14.35
12	Tomodachi Life	2013.0	3DS	5.15
13	Luigi's Mansion: Dark Moon	2013.0	3DS	4.58
14	Super Mario 3D World	2013.0	WiiU	4.25
15	The Legend of Zelda: A Link Between Worlds	2013.0	3DS	3.07
16	New Super Mario Bros. 2	2012.0	3DS	9.82
17	Animal Crossing: New Leaf	2012.0	3DS	9.09
18	Pokemon Black 2/Pokemon White 2	2012.0	DS	8.33

Рисунок 9. Результат третьего запроса

Запрос 4: Продажи по платформам.

```sql
SELECT
    Platform,
    COUNT(*) as Games_Released,
    ROUND(SUM(Global_Sales), 2) as Total_Sales
FROM video_games
WHERE Platform IS NOT NULL
GROUP BY Platform
HAVING Games_Released >= 20
ORDER BY Total_Sales DESC
LIMIT 10;
```

1	Platform	Games_Released	Total_Sales
2	Wii	40	444.46
3	X360	64	391.59
4	PS2	82	390.45
5	PS3	56	333.88
6	DS	39	312.75
7	PS	45	221.63
8	PS4	27	141.09

Рисунок 10. Результат четвертого запроса

Запрос 5: Динамика продаж по годам.

```
SELECT
    Year,
    COUNT(*) AS Releases,
    ROUND(SUM(Global_Sales), 2) AS Total_Sales_M,
    ROUND(AVG(Global_Sales), 3) AS Avg_Sales_M
FROM video_ games
WHERE Year IS NOT NULL
    AND Year BETWEEN 1990 AND 2016
GROUP BY Year
ORDER BY Year;
```

1	Year	Releases	Total_Sales	Avg_Sales
2	1990.0	4	31.99	7.9975
3	1991.0	2	8.95	4.475
4	1992.0	7	45.33	6.475714285714285
5	1993.0	2	13.540000000000001	6.7700000000000005
6	1994.0	6	28.5	4.75
7	1995.0	8	29.98	3.7475
8	1996.0	12	94.76	7.896666666666667
9	1997.0	11	65.87	5.988181818181818
10	1998.0	13	73.97	5.6899999999999995
11	1999.0	17	105.39	6.199411764705882
12	2000.0	14	56.5	4.035714285714286
13	2001.0	24	125.81	5.242083333333333
14	2002.0	24	118.61	4.942083333333334
15	2003.0	14	61.1	4.364285714285715

Рисунок 11. Результат пятого запроса

8. Экспорт результатов.

```
import sqlite3
import csv
import json
from pathlib import Path

print("Экспорт данных из базы данных в CSV и JSON")
print("=" * 50)

# Проверяем базу данных
db_path = Path("vgsales.db")
if not db_path.exists():
    print("Ошибка: База данных 'vgsales.db' не найдена")
    print("Сначала запустите: python create_db.py")
    exit(1)

print(f"База данных найдена: {db_path}")

# Создаем папки для результатов
results_dir = Path("results")
csv_dir = results_dir / "csv"
json_dir = results_dir / "json"

csv_dir.mkdir(parents=True, exist_ok=True)
json_dir.mkdir(parents=True, exist_ok=True)

print("Папки для результатов созданы")

# Подключаемся к базе данных
conn = sqlite3.connect('vgsales.db')
conn.row_factory = sqlite3.Row
cursor = conn.cursor()

print("Подключение к базе данных установлено")

# Читаем SQL запросы из файла
sql_file = Path("sql_scripts/analysis.sql")
if not sql_file.exists():
    print(f"Ошибка: Файл {sql_file} не найден")
    conn.close()
    exit(1)

with open(sql_file, 'r', encoding='utf-8') as f:
    sql_content = f.read()

# Разделяем запросы
queries = []
current_query = ""
```

```
for line in sql_content.split('\n'):
    line = line.strip()
    if line.startswith('--') or not line:
        continue

    current_query += line + " "
    if ';' in line:
        query = current_query.strip()
        if query and len(query) > 10:
            queries.append(query)
        current_query = ""

if current_query and len(current_query.strip()) > 10:
    queries.append(current_query.strip())

print(f"Найдено {len(queries)} SQL запросов")

# Выполняем каждый запрос и экспортируем результаты
exported_count = 0
for i, query in enumerate(queries, 1):
    if not query:
        continue

    try:
        # Выполняем запрос
        cursor.execute(query)
        results = cursor.fetchall()

        if not results:
            print(f"Запрос {i}: Нет результатов")
            continue

        # Получаем названия колонок
        column_names = [description[0] for description in cursor.description]

        print(f"Запрос {i}: {len(results)} записей")

        # Экспорт в CSV
        csv_filename = csv_dir / f"query_{i:02d}.csv"
        with open(csv_filename, 'w', newline='', encoding='utf-8') as csv_file:
            writer = csv.writer(csv_file)
            writer.writerow(column_names)
            for row in results:
                writer.writerow(row)

        # Экспорт в JSON
        json_filename = json_dir / f"query_{i:02d}.json"
        data_for_json = []
        for row in results:
            data_for_json.append(dict(zip(column_names, row)))
```

```
            with open(json_filename, 'w', encoding='utf-8') as json_file:
                json.dump(data_for_json, json_file, ensure_ascii=False, indent=2)

            print(f"  Экспортировано: {csv_filename.name}, {json_filename.name}")

            exported_count += 1

        except sqlite3.Error as e:
            print(f"Ошибка SQL в запросе {i}: {e}")
            continue
        except Exception as e:
            print(f"Ошибка в запросе {i}: {str(e)[:100]}")
            continue

# Закрываем соединение
conn.close()

print("\nЭкспорт завершен")
print(f"Обработано запросов: {len(queries)}")
print(f"Успешно экспортировано: {exported_count}")

if exported_count > 0:
    print("\nСозданные файлы:")
    for csv_file in sorted(csv_dir.glob('*.csv')):
        print(f"  {csv_file.name}")
else:
    print("Не экспортировано ни одного файла")
```

Вывод: в ходе выполнения лабораторной работы были исследованы базовые возможности системы управления базами данных SQLite3.