Федеральное государственное автономное образовательное учреждение высшего образования

«Национальный исследовательский университет ИТМО»

Факультет программной инженерии и компьютерной техники

Лабораторная работа №4

Информатика

Введение в базы данных

Выполнила: Гафурова Фарангиз Фуркатовна

Группа: P3120

Преподаватель: Болдырева Елена Александровна

г. Санкт—Петербург

2023

Оглавление

Текст задания:

1. Самостоятельно выберите тематику для своей собственной базы данных. База данных должна содержать не менее четырех таблиц. Каждая таблица должна иметь минимум одну связь с другими таблицами для формирования сложных запросов. Каждая таблица содержит не менее 5 записей.
2. Нарисуйте схему базы данных как в примере с указанием связей между таблицами.
3. Создайте таблицы с помощью Python для SQLite и MySQL.
4. Добавьте по одной новой записи в каждую из таблиц Вашей базы данных.
5. Продемонстрируйте работу запросов на извлечение данных:

   - выбрать все записи из таблиц,

   - составить запрос по извлечению данных с использованием JOIN

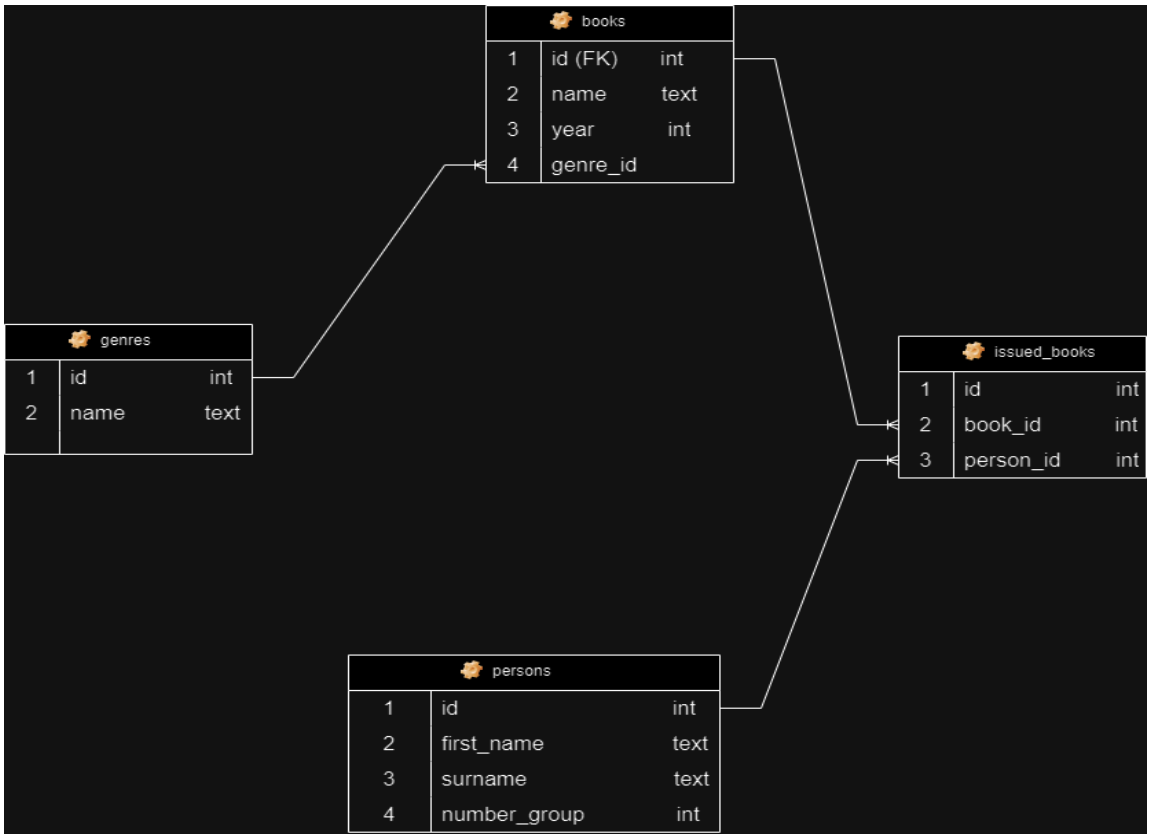
   - составить запрос по извлечению данных с использованием WHERE и GROUP BY

   Составить два запроса, в которых будет вложенный SELECT-запрос (вложение с помощью WHERE.

   - составить 2 запроса с использованием UNION (объединение запросов).

   Составить 1 запрос с использованием DISTINCT. Если для демонстрации работы этого ключевого слова недостаточно данных – предварительно дополните таблицу.
6. Обновить две записи в двух разных таблицах Вашей базы данных
7. Удалить по одной записи из каждой таблицы. Удалите все записи в одной из таблиц.

Схема базы данных



Основные этапы вычисления

Код для SQLite

1. Подключение к базам данных SQLite

```
import sqlite3
from sqlite3 import Error

def create_connection(path):
    connection = None
    try:
        connection = sqlite3.connect(path)
        print("Connection to SQLite DB successful")
    except Error as e:
        print(f"The error '{e}' occurred")

    return connection
```

2. Создание таблиц books, issued_books, persons, genres

```
def execute_query(connection, query, fetch_result=False):
    cursor = connection.cursor()
    try:
        cursor.execute(query)
        connection.commit()
        if fetch_result:
            return cursor.fetchall()
    except sqlite3.Error as e:
        print(f"The error '{e}' occurred")
connection = create_connection("library.sqlite")
create_persons_table = """
CREATE TABLE IF NOT EXISTS persons (
  id INTEGER PRIMARY KEY AUTOINCREMENT,
  first_name TEXT NOT NULL,
  surname TEXT NOT NULL,
  number_group INTEGER
);
"""
execute_query(connection, create_persons_table)

create_genres_table = """
CREATE TABLE IF NOT EXISTS genres (
  id INTEGER PRIMARY KEY AUTOINCREMENT,
  name TEXT NOT NULL
);
"""
execute_query(connection, create_genres_table)

create_books_table = """
CREATE TABLE IF NOT EXISTS books (
  id INTEGER PRIMARY KEY AUTOINCREMENT,
  name TEXT NOT NULL,
  year INTEGER NOT NULL,
  genre_id INTEGER NOT NULL,
  FOREIGN KEY (genre_id) REFERENCES genres (id)
);
"""
execute_query(connection, create_books_table)

create_issued_books_table = """
CREATE TABLE IF NOT EXISTS issued_books (
  id INTEGER PRIMARY KEY AUTOINCREMENT,
```

```
  book_id INTEGER NOT NULL,
  person_id INTEGER NOT NULL,
  FOREIGN KEY (book_id) REFERENCES books (id),
  FOREIGN KEY (person_id) REFERENCES persons (id)
);
"""
execute_query(connection, create_issued_books_table)
```

3. Вставка записей в таблицы

```
create_persons = """
INSERT INTO
  persons (first_name, surname, number_group)
VALUES
  ('James', 'Hype', 1),
  ('Leila', 'Corr', 1),
  ('Brigitte', 'Blue', 2),
  ('Mike', 'Cin', 2),
  ('Elizabeth', 'Viola', 3)
"""
execute_query(connection, create_persons)

create_books = """
INSERT INTO
  books (name, year, genre_id)
VALUES
  ('Crazy Town', 2010, 1),
  ('Charlie Gehringer', 2013, 2),
  ('Childbed Fever', 2015, 3),
  ('Hitler Made Me', 2010, 2),
  ('My Dirty Little', 2020, 5)
"""
execute_query(connection, create_books)

create_genres = """
INSERT INTO
  genres (name)
VALUES
  ('Classic'),
  ('Mystic'),
  ('Fantasy'),
  ('Detective'),
  ('Comedy')
"""
execute_query(connection, create_genres)

create_issued_books = """
INSERT INTO
  issued_books (book_id, person_id)
VALUES
  (1, 1),
  (1, 2),
  (2, 1),
  (2, 3),
  (3, 4),
  (4, 4),
  (5, 5)
"""
execute_query(connection, create_issued_books)
```

4. Извлечение данных из записей

```
select_all_persons = """
SELECT * FROM persons;
"""
```

```
print('1.', execute_query(connection, select_all_persons, fetch_result=True))

select_all_genres = """
SELECT * FROM genres;
"""

print('2.', execute_query(connection,select_all_genres,fetch_result=True))

select_all_books = """
SELECT * FROM books;
"""

print('3.', execute_query(connection,select_all_books,fetch_result=True))

select_all_issued_books = """
SELECT * FROM issued_books;
"""

print('4.',
execute_query(connection,select_all_issued_books,fetch_result=True))
```

5. Извлечение данных с помощью JOIN

```
select_issued_books_info = """
SELECT persons.first_name, persons.surname, books.name, books.year
FROM issued_books
JOIN persons ON issued_books.person_id = persons.id
JOIN books ON issued_books.book_id = books.id;
"""
print('2.', execute_query(connection, select_issued_books_info, True))
```

6. Извлечение данных с использование WHERE и GROUP BY

```
select_books_count_by_genre = """
SELECT genres.name, COUNT(books.id) as book_count
FROM books
JOIN genres ON books.genre_id = genres.id
WHERE books.year > 2010
GROUP BY genres.name;
"""
print('3.', execute_query(connection, select_books_count_by_genre, True))
```

7. Два запроса с вложенным SELECT-запросом (вложение с помощью WHERE)

```
select_authors_of_books_after_2010 = """
SELECT first_name, surname
FROM persons
WHERE id IN (SELECT person_id FROM issued_books
             JOIN books ON issued_books.book_id = books.id
             WHERE books.year > 2010);
"""
print('4.', execute_query(connection, select_authors_of_books_after_2010,
True))
```

8. Два запроса с использование UNION (объединение запросов)

```
union_query = """
SELECT first_name, surname FROM persons WHERE number_group = 1
UNION
SELECT first_name, surname FROM persons WHERE number_group = 2
"""
print('5.', execute_query(connection, union_query, fetch_result=True))
```

9. Запрос с использованием DISTINCT

```
select_distinct_genres = """
SELECT name FROM genres;
"""

print('6.', execute_query(connection, select_distinct_genres,
fetch_result=True))
```

10. Обновление двух записей в двух разных таблицах

```
update_person_name = """
UPDATE persons SET first_name = 'John' WHERE id = 1;
"""
update_book_year = """
UPDATE books SET year = 2012 WHERE id = 1;
"""
execute_query(connection, update_person_name)
execute_query(connection, update_book_year)
```

11. Удаление по одной записи из каждой таблицы

```
delete_person = """
DELETE FROM persons WHERE id = 2;
"""
delete_book = """
DELETE FROM books WHERE id = 3;
"""
execute_query(connection, delete_person)
execute_query(connection, delete_book
```

12. Удаление всех записей в одной из таблиц

```
delete_all_genres = """
DELETE FROM genres;
"""
execute_query(connection, delete_all_genres)
```

13. Форматирование отчета

```
report_query = """
SELECT persons.first_name, persons.surname, GROUP_CONCAT(books.name, ', ') AS
issued_books
FROM issued_books
JOIN persons ON issued_books.person_id = persons.id
JOIN books ON issued_books.book_id = books.id
GROUP BY persons.first_name, persons.surname;
"""
execute_query(connection, report_query)
```

14. Закрываем соединение с базой данных

```
if connection:
    connection.close()
```

15. Вывод программы

```
C:\Users\ПК\PycharmProjects\Infa_Laba4\.venv\Scripts\python.exe C:\Users\ПК\PycharmProjects\Infa_Laba4\main.py
Connection to SQLite DB successful
1. [(1, 'James', 'Hype', 1), (2, 'Leila', 'Corr', 1), (3, 'Brigitte', 'Blue', 2), (4, 'Mike', 'Cin', 2), (5, 'Elizabeth', 'Viola', 3)]
2. [(1, 'Classic'), (2, 'Mystic'), (3, 'Fantasy'), (4, 'Detective'), (5, 'Comedy')]
3. [(1, 'Crazy Town', 2010, 1), (2, 'Charlie Gehringer', 2013, 2), (3, 'Childbed Fever', 2015, 3), (4, 'Hitler Made Me', 2010, 2), (5, 'My Dirty Little', 2020, 5)]
4. [(1, 1, 1), (2, 1, 2), (3, 2, 1), (4, 2, 3), (5, 3, 4), (6, 4, 4), (7, 5, 5)]
5. [('James', 'Hype', 'Crazy Town', 2010), ('Leila', 'Corr', 'Crazy Town', 2010), ('James', 'Hype', 'Charlie Gehringer', 2013), ('Brigitte', 'Blue', 'Charlie Gehringer', 2013), (
6. [('Comedy', 1), ('Fantasy', 1), ('Mystic', 1)]
7. [('James', 'Hype'), ('Brigitte', 'Blue'), ('Mike', 'Cin'), ('Elizabeth', 'Viola')]
8. [('Brigitte', 'Blue'), ('James', 'Hype'), ('Leila', 'Corr'), ('Mike', 'Cin')]
9. [('Classic',), ('Mystic',), ('Fantasy',), ('Detective',), ('Comedy',)]

Process finished with exit code 0
```

Код для MySQL

1. Подключение, создание таблиц, вставка записей в MySQL

```python
import mysql.connector

def create_connection():
    connection = None
    try:
        connection = mysql.connector.connect(
            host="localhost",
            user="root",
            password="Miranda2006.",
            database="world"
        )
        print("Соединение с базой данных установлено")
    except Exception as e:
        print(f"Ошибка при подключении к базе данных: {e}")
    return connection

def execute_query(connection, query, fetch_result=False):
    cursor = connection.cursor()
    try:
        cursor.execute(query)
        if fetch_result:
            result = cursor.fetchall()
            return result
    except mysql.connector.Error as e:
        print(f"Ошибка при выполнении запроса: {e}")
    finally:
        cursor.close()

    # Вызываем commit() только если запрос изменяет данные
    if query.strip().lower().startswith(("insert", "update", "delete")):
        connection.commit()

# Создание соединения
connection = create_connection()

# Определение запросов
create_persons_table = """
CREATE TABLE IF NOT EXISTS persons (
  id INT AUTO_INCREMENT PRIMARY KEY,
  first_name VARCHAR(255) NOT NULL,
  surname VARCHAR(255) NOT NULL,
  number_group INT
)
"""
create_genres_table = """
CREATE TABLE IF NOT EXISTS genres (
  id INT AUTO_INCREMENT PRIMARY KEY,
  name VARCHAR(255) NOT NULL
)
"""
create_books_table = """
CREATE TABLE IF NOT EXISTS books (
  id INT AUTO_INCREMENT PRIMARY KEY,
  name VARCHAR(255) NOT NULL,
  year INT NOT NULL,
  genre_id INT NOT NULL,
  FOREIGN KEY (genre_id) REFERENCES genres (id) ON DELETE CASCADE
)
"""
create_issued_books_table = """
CREATE TABLE IF NOT EXISTS issued_books (
  id INT AUTO_INCREMENT PRIMARY KEY,
  book_id INT NOT NULL,
  person_id INT NOT NULL,
```

```
  FOREIGN KEY (book_id) REFERENCES books (id) ON DELETE CASCADE,
  FOREIGN KEY (person_id) REFERENCES persons (id)  ON DELETE CASCADE
)
"""
create_persons = """
INSERT INTO persons (first_name, surname, number_group)
VALUES
  ('James', 'Hype', 1),
  ('Leila', 'Corr', 1),
  ('Brigitte', 'Blue', 2),
  ('Mike', 'Cin', 2),
  ('Elizabeth', 'Viola', 3)
"""
create_books = """
INSERT INTO books (name, year, genre_id)
VALUES
  ('Crazy Town', 2010, 1),
  ('Charlie Gehringer', 2013, 2),
  ('Childbed Fever', 2015, 3),
  ('Hitler Made Me', 2010, 2),
  ('My Dirty Little', 2020, 5)
"""
create_genres = """
INSERT INTO genres (name)
VALUES
  ('Classic'),
  ('Mystic'),
  ('Fantasy'),
  ('Detective'),
  ('Comedy')
"""
create_issued_books = """
INSERT INTO issued_books (book_id, person_id)
VALUES
  (1, 1),
  (1, 2),
  (2, 1),
  (2, 3),
  (3, 4),
  (4, 4),
  (5, 5)
"""

# Выполнение запросов
execute_query(connection, create_persons_table)
execute_query(connection, create_genres_table)
execute_query(connection, create_books_table)
execute_query(connection, create_issued_books_table)

execute_query(connection, create_genres)
execute_query(connection, create_persons)
execute_query(connection, create_books)
execute_query(connection, create_issued_books)
```

2. Извлечение данных с помощью JOIN

```
select_issued_books_info = """
SELECT persons.first_name, persons.surname, books.name, books.year
FROM issued_books
JOIN persons ON issued_books.person_id = persons.id
JOIN books ON issued_books.book_id = books.id;
"""
print('5.', execute_query(connection, select_issued_books_info, True))
```

3. Извлечение данных с использованием WHERE и GROUP BY

```
select_books_count_by_genre = """
SELECT genres.name, COUNT(books.id) as book_count
FROM books
JOIN genres ON books.genre_id = genres.id
WHERE books.year > 2010
GROUP BY genres.name;
"""
print('6.', execute_query(connection, select_books_count_by_genre, True))
```

4. Два запроса с вложенным SELECT-запросом (вложение с помощью WHERE)

```
select_authors_of_books_after_2010 = """
SELECT first_name, surname
FROM persons
WHERE id IN (SELECT person_id FROM issued_books
             JOIN books ON issued_books.book_id = books.id
             WHERE books.year > 2010);
"""
print('7.', execute_query(connection, select_authors_of_books_after_2010,
True))
```

5. Два запроса с использование UNION (объединение запросов)

```
union_query = """
SELECT first_name, surname FROM persons WHERE number_group = 1
UNION
SELECT first_name, surname FROM persons WHERE number_group = 2
"""
print('8.', execute_query(connection, union_query, fetch_result=True))
```

6. Запрос с использованием DISTINCT

```
select_distinct_genres = """
SELECT name FROM genres;
"""

print('9.', execute_query(connection, select_distinct_genres,
fetch_result=True))
```

7. Обновление двух записей в двух разных таблицах

```
update_person_name = """
UPDATE persons SET first_name = 'John' WHERE id = 1;
"""
update_book_year = """
UPDATE books SET year = 2012 WHERE id = 1;
"""
execute_query(connection, update_person_name)
execute_query(connection, update_book_year)
```

8. Удаление по одной записи из каждой таблицы

```
delete_person = """
DELETE FROM persons WHERE id = 2;
"""
delete_book = """
DELETE FROM books WHERE id = 3;
"""
execute_query(connection, delete_person)
execute_query(connection, delete_book)
```

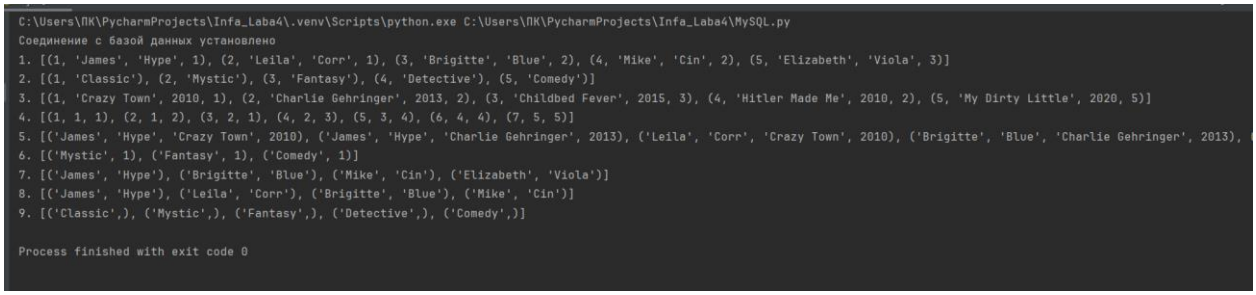
9. Удаление всех записей в одной из таблиц

```
delete_all_genres = """
DELETE FROM genres;
"""
execute_query(connection, delete_all_genres)
```

10.Закрываем соединение с базой данных

```
if connection:
    connection.close()
```

11.Вывод программы



Вывод

В ходе данной лабораторной работы я поняла, как работать с базами данных таких как: SQLite и MySQL. Так же выяснила что SQLite и MySQL — это две различные базы данных с отличающимися характеристиками и функциональностью. SQLite является легковесной встроенной базой данных, которая хранит всю информацию в одном файле и подходит для маленьких проектов или приложений с низкой производительностью. С другой стороны, MySQL — это более мощная и распространенная база данных, использующая серверную архитектуру, которая поддерживает большое количество запросов и может обрабатывать большие объемы данных. Обе базы данных поддерживают SQL-запросы, но MySQL предоставляет больше функций и возможностей. Выбор между SQLite и MySQL зависит от конкретных потребностей проекта.