НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО

Факультет ПИиКТ

Информатика

Лабораторная работа №4

Выполнил студент

Набокова Алиса Владиславовна

Преподаватель: Болдырева Елена Александровна

г. Санкт-Петербург

2023

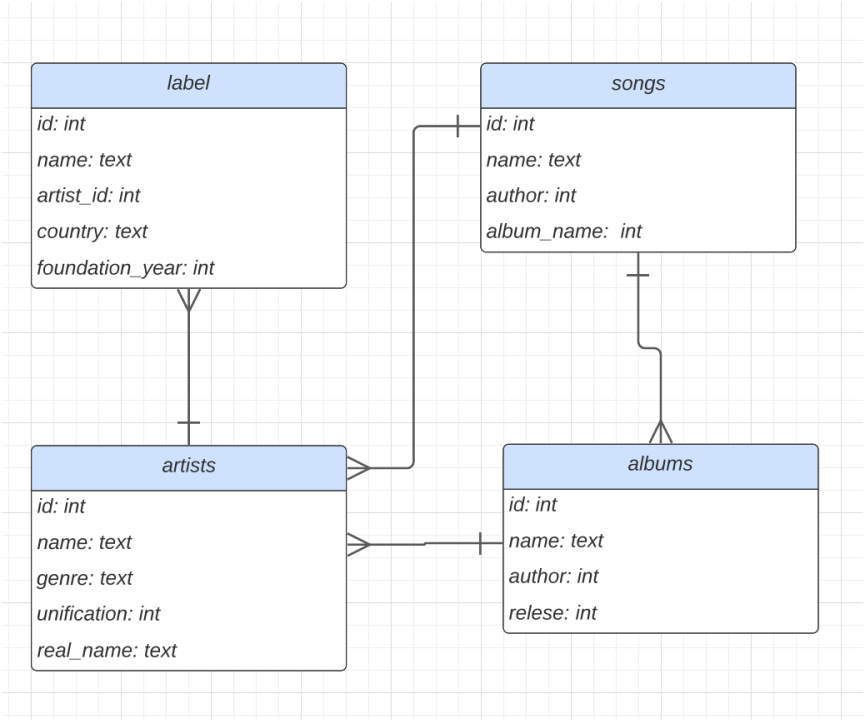
Оглавление

Задание

1. Проделайте все шаги из примера выше. Удостоверьтесь, что Вы поняли, как это работает.
2. 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.
3. Нарисуйте схему базы данных как в примере с указанием связей между таблицами.
4. Создайте таблицы с помощью Python для SQLite и MySQL.
5. Добавьте по одной новой записи в каждую из таблиц Вашей базы данных.
6. Продемонстрируйте работу запросов на извлечение данных:
7. - выбрать все записи из таблиц,
8. - составить запрос по извлечению данных с использованием JOIN
9. - составить запрос по извлечению данных с использованием WHERE и GROUP BY
10. Составить два запроса, в которых будет вложенный SELECT-запрос (вложение с помощью WHERE.
11. - составить 2 запроса с использованием UNION (объединение запросов).
12. Составить 1 запрос с использованием DISTINCT. Если для демонстрации работы этого ключевого слова недостаточно данных – предварительно дополните таблицу.
13. 7. Обновить две записи в двух разных таблицах Вашей базы данных
14. 8. Удалить по одной записи из каждой таблицы.
15. Удалите все записи в одной из таблиц.
16. Сформируйте отчет.

Отчет

Схема базы данных



Подключение к sqlite

```python
import sqlite3
from sqlite3 import Error
#подключение к БД
1 usage
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}' occured")

    return connection
connection = create_connection("/Users/alyssanabokova/Documents/лабы/инфа/Laba4/sm_app.sqlite")

```

Создание таблиц

```python
15 usages
def execute_query(connection, query):
    cursor = connection.cursor()
    try:
        cursor.execute(query)
        connection.commit()
        print("Query executed successfully")
    except Error as e:
        print(f"The error '{e}' occurred")

```

```python
#label
create_label_table = """
CREATE TABLE IF NOT EXISTS label (
    id INTEGER PRIMARY KEY AUTOINCREMENT,
    name TEXT NOT NULL,
    artist_name TEXT NOT NULL,
    country TEXT,
    foundation_year INTEGER,
    FOREIGN KEY (artist_name) REFERENCES artists (name)

);
"""
execute_query(connection, create_label_table)

#artists
create_artists_table = """
CREATE TABLE IF NOT EXISTS artists(
    id INTEGER PRIMARY KEY AUTOINCREMENT,
    name TEXT NOT NULL,
    genre TEXT NOT NULL,
    unification TEXT NOT NULL,
    real_name TEXT,
    FOREIGN KEY (unification) REFERENCES label (name)
);
"""
execute_query(connection, create_artists_table)
```

```python
#albums
create_albums_table = """
CREATE TABLE IF NOT EXISTS albums (
     id INTEGER PRIMARY KEY AUTOINCREMENT,
     name TEXT NOT NULL,
     author TEXT NOT NULL,
     relese INTEGER NOT NULL,
     FOREIGN KEY (author) REFERENCES artists (name)

);
"""
execute_query(connection, create_albums_table)

#songs
create_songs_table = """
CREATE TABLE IF NOT EXISTS songs (
    id INTEGER PRIMARY KEY AUTOINCREMENT,
    name TEXT NOT NULL,
    author TEXT NOT NULL,
    album_name TEXT NOT NULL,
    FOREIGN KEY (author) REFERENCES artists (name)
    FOREIGN KEY (album_name) REFERENCES albums (name)

);
"""
execute_query(connection, create_songs_table)
```

Заполнение таблиц

```
#в таблицу label
create_label = """
INSERT INTO
    label (name, artist_name, country, foundation_year)
VALUES
    ('Dead Dynasty', 'SALUKI, Pharaoh, Boulevard Depo', 'Russia', 2013);
"""
execute_query(connection, create_label)

#в таблицу artists
create_artists = """
INSERT INTO
    artists (name, genre, unification, real_name)
VALUES
    ('SALUKI', 'alternative', 'Dead Dynasty', 'Arseniy'),
    ('Pharaoh', 'hip-hop', 'Dead Dynasty', 'Gleb'),
    ('Boulevard Depo', 'hip-hop/rap', 'Dead Dynasty', 'Artem');
"""
execute_query(connection, create_artists)
```

```
#в таблицу albums
create_albums = """
INSERT INTO
    albums (name, author, relese)
VALUES
    ('Wild East', 'SALUKI', 2023),
    ('Pink Phloyd', 'Pharaoh', 2017),
    ('Rapp2', 'Boulevard Depo', 2018);
"""
execute_query(connection, create_albums)

#в таблице songs
create_songs = """
INSERT INTO
    songs (name, author, album_name)
VALUES
    ('MAGDALENE', 'SALUKI', 'Wild East'),
    ('Lallipap', 'Pharaoh', 'Pink Phloyd'),
    ('White Trash', 'Boulevard Depo', 'Rapp2');
"""
execute_query(connection, create_songs)
```

Извлечение данных из записи

```python
#извлечение данных из записи
8 usages
def execute_read_query(connection, query):
    cursor = connection.cursor()
    result = None
    try:
        cursor.execute(query)
        result = cursor.fetchall()
        return result
    except Error as e:
        print(f"The error '{e}' occurred")

#выберем все записи из таблицы songs
select_songs = "SELECT * from songs"
songs = execute_read_query(connection, select_songs)

for song in songs:
    print(song)
```

Вывод

```
(7, 'MAGDALENE', 'SALUKI', 'Wild East')
(8, 'Lallipap', 'Pharaoh', 'Pink Phloyd')
(9, 'White Trash', 'Boulevard Depo', 'Rapp2')
```

```python
#составление запроса по извлечении данных с использованием JOIN
#возвращает название песни и имя автора
select_songs = """
SELECT
   songs.author,
   songs.name,
   albums.name
FROM
    songs
    INNER JOIN albums ON albums.name = songs.album_name
"""
songs_albums = execute_read_query(connection,select_songs)

for songs_album in songs_albums:
    print(songs_album)
```

Вывод

```
('SALUKI', 'MAGDALENE', 'Wild East')
```

```
#запрос по извлечению данных с использованием WHERE и GROUP BY
#вывод песни, альбома, в который она входит и год релиза
select_relese_albums_min = """
SELECT
    songs.name,
    songs.album_name,
    albums.relese
FROM
    songs,
    albums
WHERE
    albums.name = songs.album_name
GROUP BY
    songs.album_name
"""
relese_albums_min = execute_read_query(connection,select_relese_albums_min)

for relese_album_min in relese_albums_min:
    print(relese_album_min)

```

Вывод

```
('Lallipap', 'Pink Phloyd', 2016)
('White Trash', 'Rapp2', 2018)
('MAGDALENE', 'Wild East', 2023)
```

```
#вложенный SELECT-запрос по извлечению данных с ипользовани WHERE
#1 самый ранний релиз альбома
select_albums_releses_min = """
SELECT
    author, name, relese
FROM
    albums
WHERE relese = (
    SELECT MIN(relese) FROM albums
    );
"""
albums_releses_min = execute_read_query(connection, select_albums_releses_min)

for albums_relese_min in albums_releses_min:
    print(albums_relese_min)

```

Вывод

```
('Pharaoh', 'Pink Phloyd', 2016)
('Pharaoh', 'Pink Phloyd', 2016)
```

```python
#2 самый поздний релиз альбома
select_albums_releses_max = """
SELECT
    author, name, relese
FROM
    albums
WHERE relese = (
    SELECT MAX(relese) FROM albums
    );
"""
albums_releses_max = execute_read_query(connection, select_albums_releses_max)

for albums_relese_max in albums_releses_max:
    print(albums_relese_max)
```

Вывод

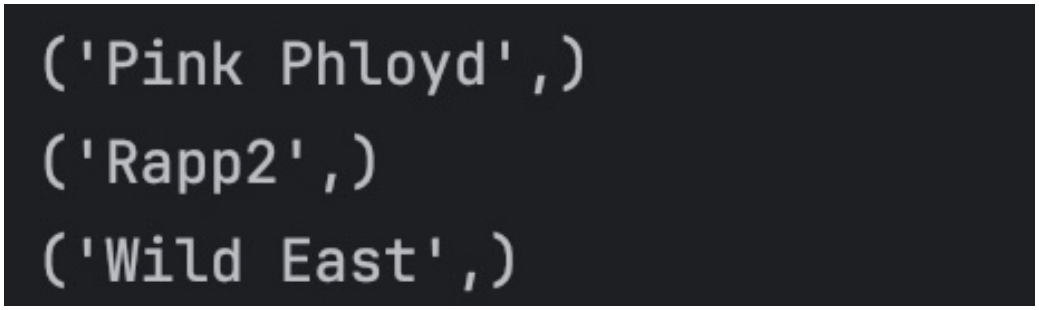
```
('SALUKI', 'Wild East', 2023)
('SALUKI', 'Wild East', 2023)
```

```python
#запросы с использованием UNION (объединённые запросы)
#1 объединяет автора альбома и песни
select_union_authors = """
SELECT author FROM songs
UNION
SELECT author FROM albums
ORDER BY author;
"""
select_union_authors = execute_read_query(connection, select_union_authors)

for select_union_author in select_union_authors:
    print(select_union_author)

#2 объединяет название альбома и альбом, в который входит песня
select_union_albums = """
SELECT name FROM albums
UNION
SELECT album_name FROM songs
ORDER BY album_name;
"""
select_union_albums = execute_read_query(connection, select_union_albums)

for select_union_album in select_union_albums:
    print(select_union_album)
```

Вывод#1

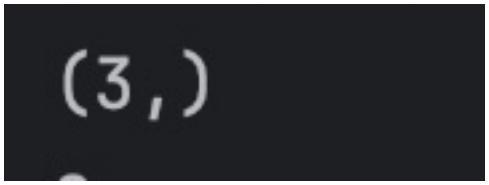
```
('Boulevard Depo',)
('Pharaoh',)
('SALUKI',)
```

Вывод#2




```
#количество уникальных имён исполнителей
select_distinct_artists = """
SELECT
    COUNT(DISTINCT real_name)
FROM
    artists;
"""
select_distinct_artists = execute_read_query(connection, select_distinct_artists)

for select_distinct_artist in select_distinct_artists:
    print(select_distinct_artist)
```

Вывод



Обновление записей таблицы

```
#обновление записей
#в таблице albums
update_album_releses = """
UPDATE
    albums
SET
    relese = 2016
WHERE
    id = 2
"""
execute_query(connection,update_album_releses)
```

```
#в таблице songs
update_songs_name = """
UPDATE
    songs
SET
     name = "Friendly Fire"
WHERE
    id = 3
"""
execute_query(connection,update_songs_name)

```

Удаление записей из таблицы

```
#удаление записей
delete_label = "DELETE FROM label WHERE id = 1"
execute_query(connection, delete_label)

delete_artists = "DELETE FROM artists WHERE id = 2"
execute_query(connection, delete_artists)

delete_albums = "DELETE FROM albums WHERE id = 3"
execute_query(connection, delete_albums)

delete_songs = "DELETE FROM songs WHERE id = 2"
execute_query(connection, delete_songs)

delete_all = "DELETE FROM songs"
execute_query(connection, delete_all)

```

Вывод:

В ходе данной лабораторной работы я узнала основы баз данных и научилась работать с sqlite