Тема: Создание и работа с многотабличной БД

Принцип работы запросов на объединения таблиц в SQL и реляционных базах данных заключается в том, что внутри одного SQL запроса SELECT выполняется два или более подзапроса (в зависимости от того, сколько мы хотим объединить таблиц), подзапросы разделяются между собой ключевым словом JOIN. У этого JOIN есть ограничение ON, которое называют предикатом объединения. Предикат объединения – это всегда какое-то условие, с помощью которого РСУБД определяет какие строки из двух таблиц ей нужно объединять. А вот с тем, как объединять строки, SQLite разбирается специальным модификаторам: INNER, LEFT OUTER или просто LEFT и CROSS.

Предикат INNER

1. Готовим таблицы для реализации примеров SQL запросов JOIN в базе данных SQLite

```
with sq.connect('music.db') as con:
    con.execute('PRAGMA foreign_keys = ON')
    cur = con.cursor()
    cur.execute("""CREATE TABLE IF NOT EXISTS tracks(
        id INTEGER PRIMARY KEY,
        title TEXT NOT NULL,
        second INTEGER NOT NULL,
        price REAL NOT NULL,
        album_id INTEGER,
        FOREIGN KEY (album_id) REFERENCES albums(id) ON DELETE CASCADE ON UPDATE CASCADE
        )""")

with sq.connect('music.db') as con:
    cur = con.cursor()
    cur.execute("""CREATE TABLE IF NOT EXISTS albums(
        id INTEGER PRIMARY KEY,
        title TEXT NOT NULL,
        artist_id INTEGER,
        FOREIGN KEY (artist_id) REFERENCES artist(id) ON DELETE CASCADE ON UPDATE CASCADE
        )""")

with sq.connect('music.db') as con:
    cur = con.cursor()
    cur.execute("""CREATE TABLE IF NOT EXISTS artist(
        id INTEGER PRIMARY KEY,
        name TEXT NOT NULL
        )""")
```

ключевая фраза ON UPDATE CASCADE, которая указывается после FOREIGEN KEY говорит СУБД о том, что та должна модифицировать обе таблицы друг за другом – каскадно. Таким образом, указав SQLite, что данные должны модфицироваться каскадом, мы сможем написать только один SQL запрос UPDATE, а SQLite сама обновит обе таблицы.

```
con.execute('PRAGMA foreign_keys = ON')
```
разрешает работу с внешними ключами

2. Добавляем информацию в БД

create_BD.py | data.py

```python
info_tracks = [
    (1,'Девушка по городу', 193, 26.20, 2),
    (2,'Песня идущего домой', 170, 22.10, 2),
    (3,'Полковнику никто не пишет', 292, 32.15, 3),
    (4,'Мой друг', 291, 27.15, 3),
    (5,'Моё сердце', 249, 21.12, 1),
    (6,'Линия жизни', 180, 41.12, 1),
    (7,'Остаемся зимовать', 218, 17.62, 1),
]

info_artist = [
    (1, 'Вячеслав Бутусов'),
    (2, 'Сплин'),
    (3, 'Би - 2'),
]

info_albums = [
    (1, '25-й кадр', 2),
    (2, 'Биографика', 1),
    (3, 'Би - 2', 3),
]
```

3. Внутреннее объединение таблиц в базах данных SQLite: INNER JOIN в SQL

```python
#Объединение двух таблиц
with sq.connect('music.db') as con:
    cur = con.cursor()
    cur.execute("SELECT tracks.title, second, price, albums.title FROM tracks INNER JOIN "
                "Albums ON tracks.album_id = albums.id")
    result = cur.fetchall()
print(result)

#Объединение трех таблиц
with sq.connect('music.db') as con:
    cur = con.cursor()
    cur.execute("SELECT tracks.title, albums.title, artist.name FROM tracks LEFT JOIN "
                "Albums ON tracks.album_id = albums.id LEFT JOIN "
                "artist ON albums.artist_id = artist.id")
    result = cur.fetchall()
print(result)
```

4. Мы можем задавать различные условия выборки данных клаузулой WHERE в том случае, когда объединяем таблице предикатом JOIN:

SELECT tracks.title, second, price, albums.title FROM tracks INNER JOIN

Albums ON tracks.album_id = albums.id

WHERE artist_id = 2