SQL | Я.Шпора

Работа с SQLite из консоли

```
# Команда для работы с виртуальной базой (она будет удалена после выход
а из терминала):
sqlite3

# Подключение к реальной базе (в примере - db.sqlite)
sqlite3 db.sqlite
# Включить вывод результатов запросов в табличном виде:
.mode column
```

Работа с SQLite из Python

```
import sqlite3

# Если файла db.sqlite нет - он будет создан;
# будет создано соединение с базой.
con = sqlite3.connect('db.sqlite')

# Создаём объект cursor для работы с БД:
cur = con.cursor()

# Готовим SQL-запрос и сохраняем в переменную (запрос - это просто строка
query = '''
CREATE TABLE IF NOT EXISTS имя_таблицы(
    имя_столбца ТИП_СТОЛБЦА PRIMARY KEY,
    имя_столбца ТИП_СТОЛБЦА
);
'''
# Для выполнения одиночного запроса применяем метод cur.execute():
cur.execute(query)
```

```
# Закрываем соединение с БД.
con.close()
```

Создание таблиц

```
CREATE TABLE IF NOT EXISTS имя_таблицы(
    имя_столбца ТИП_СТОЛБЦА PRIMARY KEY,
    имя_столбца ТИП_СТОЛБЦА;
```

К столбцу могут быть применены ограничения:

- NOT NULL — обязательное поле
- UNIQUE — значение должно быть уникально в пределах таблицы

SQLite поддерживает пять типов данных:

- NULL — отсутствие значения;
- INTEGER — целое число;
- REAL — дробное число;
- TEXT — текст;
- BLOB — двоичные данные.

В других БД используются и другие типы.

Наполнение БД информацией из кода Python

Создаём и отправляем запрос к таблице с именем `content`, содержащей три поля:

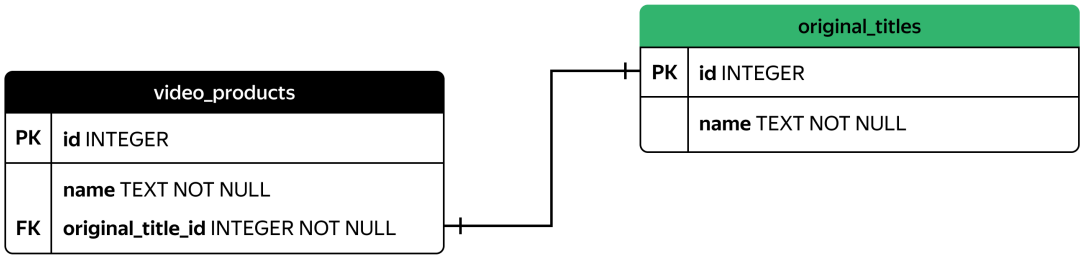
```
...
список_контента = [
    (строка_1_поле_1, строка_1_поле_2, строка_1_поле_3),
    (строка_2_поле_1, строка_2_поле_2, строка_2_поле_3),
]

# Для выполнения нескольких запросов применяем метод cur.executemany():
cur.executemany('INSERT INTO content VALUES(?, ?, ?);', список_контента)
...
```

Типы связей между таблицами реляционных БД

В связанных таблицах должно быть поле *primary key* (PK, значение, уникальное в пределах одной таблицы), а в ссылающихся таблицах — поля с внешними ключами (FK, *foreign key*), в которых будет указано, на какую запись связанной таблицы ссылается текущая запись.

Связь «один-к-одному»

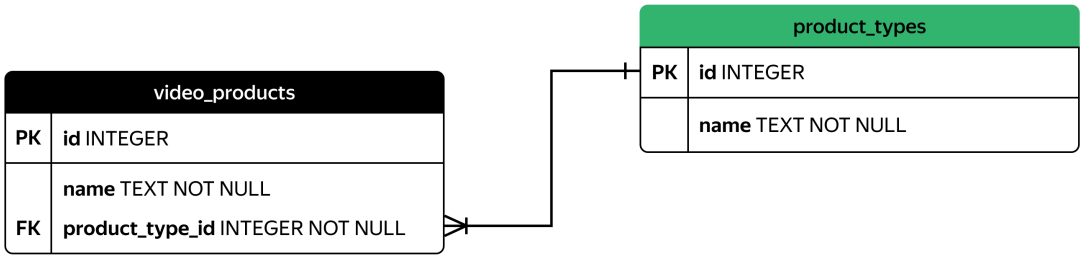
У каждого фильма в **video_products** обязательно есть оригинальное название из **original_titles**.




```sql
CREATE TABLE IF NOT EXISTS original_titles(
    id INTEGER PRIMARY KEY,
    name TEXT NOT NULL
);

CREATE TABLE IF NOT EXISTS video_products(
    id INTEGER PRIMARY KEY,
    name TEXT NOT NULL,
    original_title_id INTEGER NOT NULL UNIQUE,
    FOREIGN KEY(original_title_id) REFERENCES original_titles(id)
);
```

Связь «один-ко-многим» и «многие-к-одному»

Фильм из таблицы **video_products** может относиться только к одному типу **product_types**, а несколько фильмов могут быть одного типа; для любого фильма **обязательно** должен быть указан его тип.

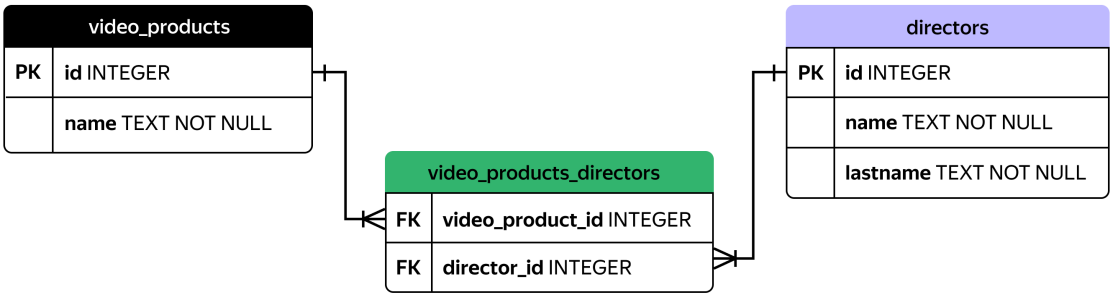



```sql
CREATE TABLE IF NOT EXISTS product_types(
    id INTEGER PRIMARY KEY,
    name TEXT NOT NULL
);
CREATE TABLE IF NOT EXISTS video_products(
    id INTEGER PRIMARY KEY,
    name TEXT NOT NULL,
    product_type_id INTEGER NOT NULL,
    FOREIGN KEY(product_type_id) REFERENCES product_types(id)
);
```

Связь «многие-ко-многим»

- У фильма может быть один или несколько режиссёров.
- Режиссёр может снять один или несколько фильмов.

Для такой связи используется промежуточная таблица:



Эту структуру можно создать с помощью таких запросов:

```
CREATE TABLE IF NOT EXISTS video_products(
    id INTEGER PRIMARY KEY,
    name TEXT NOT NULL
);

CREATE TABLE IF NOT EXISTS directors(
    id INTEGER PRIMARY KEY,
    name TEXT NOT NULL
    lastname TEXT NOT NULL
);

-- Создаём промежуточную таблицу:
CREATE TABLE IF NOT EXISTS video_products_directors(
    director_id INTEGER NOT NULL,
    video_product_id INTEGER NOT NULL,
    -- Пару полей назначаем композитным первичным ключом:
    PRIMARY KEY (director_id, video_product_id)
    FOREIGN KEY(director_id) REFERENCES directors(id),
    FOREIGN KEY(video_product_id) REFERENCES video_products(id)
);
```

Получение данных из БД

Самые востребованные операторы и ключевые слова для запросов; они должны указываться именно в таком порядке:

```
SELECT -- столбцы через запятую | обязательное поле
-- (* - для выбора всех столбцов)
-- могут применяться агрегатные функции COUNT, MIN, MAX, AVG и SUM
-- и ключевое слово DISTINCT
FROM -- таблица | обязательное поле
WHERE -- условие/фильтрация
GROUP BY -- столбец, по которому нужно сгруппировать данные
HAVING -- условие/фильтрация на уровне сгруппированных данных
ORDER BY -- столбец, по которому нужно отсортировать вывод
LIMIT -- сколько записей показывать
```

```
OFFSET -- на сколько строк сдвинуть выборку
; -- в конце запроса обязательно ставится точка с запятой
```

Запросы для получения данных

Пример запроса с **фильтрацией**:

```
SELECT name
FROM video_products
WHERE type LIKE 'Мульт%' AND release_year > 1988;
```

Пример с **ограничением** и **сдвигом** выборки:

```
SELECT name,
       lastname
FROM directors
ORDER BY lastname
LIMIT 2 OFFSET 2;
```

Агрегирующие функции:

COUNT возвращает количество строк в результирующей выборке.

MIN и **MAX** — максимальное или минимальное значения в выборке.

AVG — среднее арифметическое значение в выборке.

SUM — сумма.

Пример запроса:

```
SELECT COUNT(name)
FROM video_products
WHERE product_type_id = 1;
```

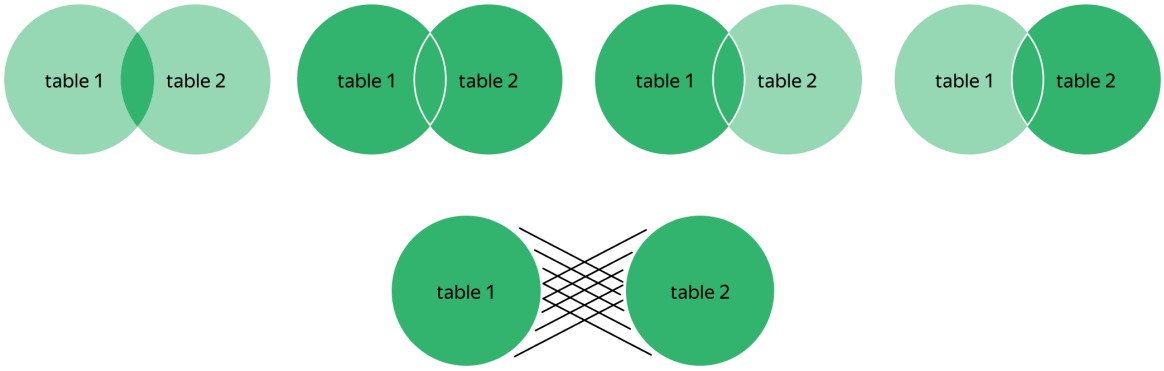
Пример запроса с **группировкой** и **фильтрацией** по сгруппированным данным:

```
SELECT name,
       COUNT(product_type_id)
FROM video_products
```

```
GROUP BY product_type_id
HAVING COUNT(product_type_id) = 1;
```

JOIN

Есть пять видов JOIN:



Пример запроса:

```
SELECT
    video_products.name,
    product_types.name
FROM
    video_products
LEFT JOIN product_types ON video_products.product_type_id = product_types
```

Изменение таблицы: ALTER TABLE

```
-- Переименование таблицы:
ALTER TABLE <имя таблицы> RENAME TO <новое имя таблицы>;

-- Добавление колонки:
ALTER TABLE <название таблицы>
ADD COLUMN <имя колонки> <тип колонки>;

-- Переименование колонки:
```

```
ALTER TABLE <название таблицы>
RENAME COLUMN <старое имя колонки> TO <новое имя колонки>;

-- Удаление колонки (в классическом SQL):
ALTER TABLE <название таблицы>
DROP COLUMN <имя колонки>;

-- Удаление таблицы:
DROP TABLE <имя таблицы>;
```