



SQL - structured query language

язык структурированных запросов

- *декларативный* язык программирования,
- применяемый для создания, модификации и управления данными в реляционной базе данных,
- управляемой соответствующей системой управления базами данных.



- встраиваемая БД,
- нет сервера,
- есть библиотека, которая устанавливается в вашу программу,
- всё (таблицы, запросы) хранятся в одном файле,
- читать один файл могут сразу несколько процессов,
- перед транзакцией записи, файл блокируется для других запросов,
- динамическая типизация.

Библиотека SQLite

- написана на C для всех платформ и многих ЯП:
Swift, Delphi, C++, Java, C#, VB.NET,
Python, Perl, Node.js, PHP, Ruby, Haskell...

Синтаксис SQL

Операторы:

- 1) Data Definition Language – определение данных
- 2) Data Manipulation Language – манипуляция данными
- 3) Data Control Language – управление доступом
- 4) Transaction Control Language – управление транзакциями

Функции:

- abs, min, trim, random, lower, length, ...

документация – [официальная](#) – [на русском](#)

SQLiteStudio

Репозиторий:

Windows files:

SQLiteStudio-3.2.1.zip (*portable*)

InstallSQLiteStudio-3.2.1.exe (*installer*)

Linux files (64 bit):

sqlitestudio-3.2.1.tar.xz (*portable*)

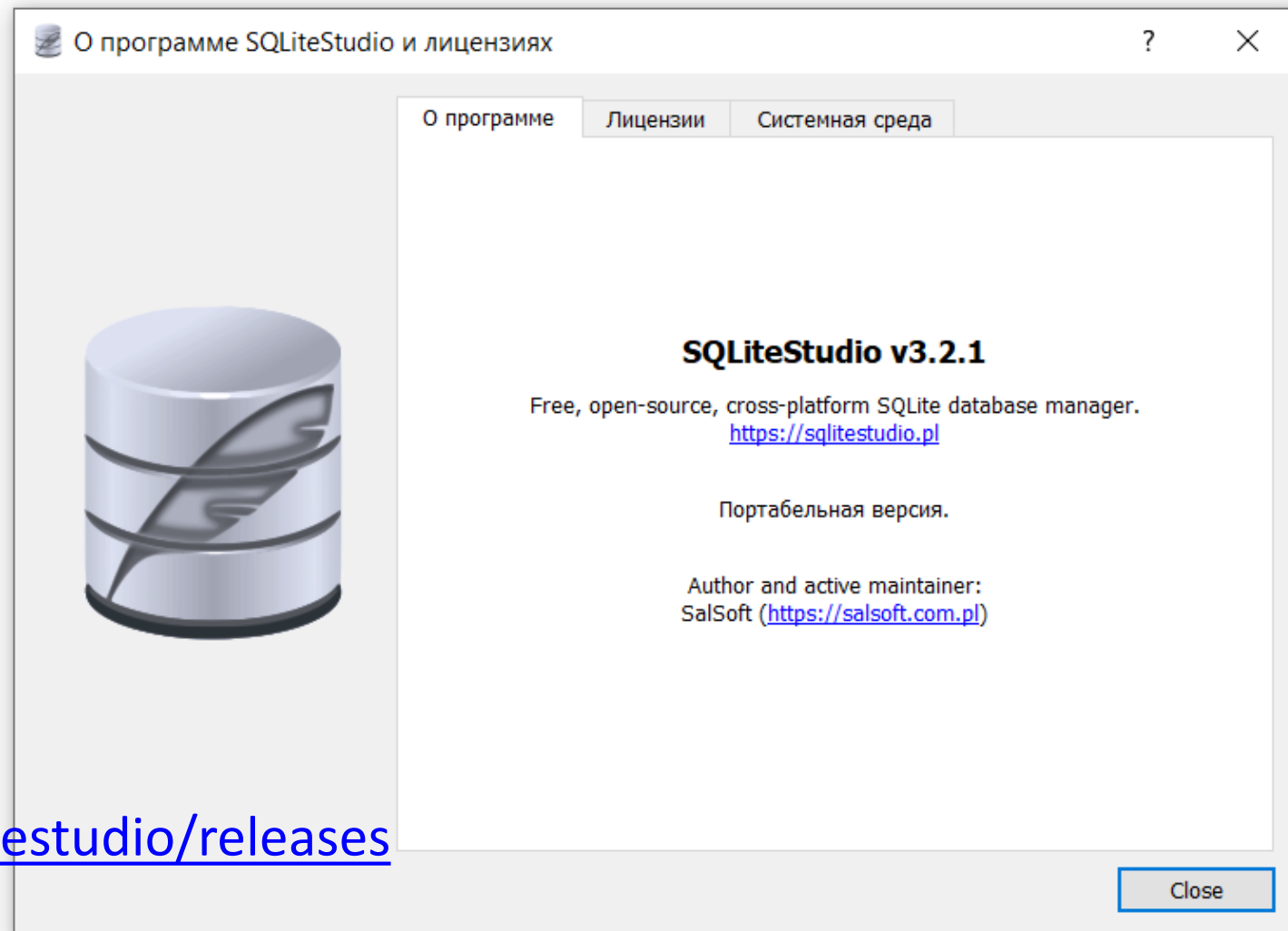
InstallSQLiteStudio-3.2.1 (*installer*)

MacOS X files:

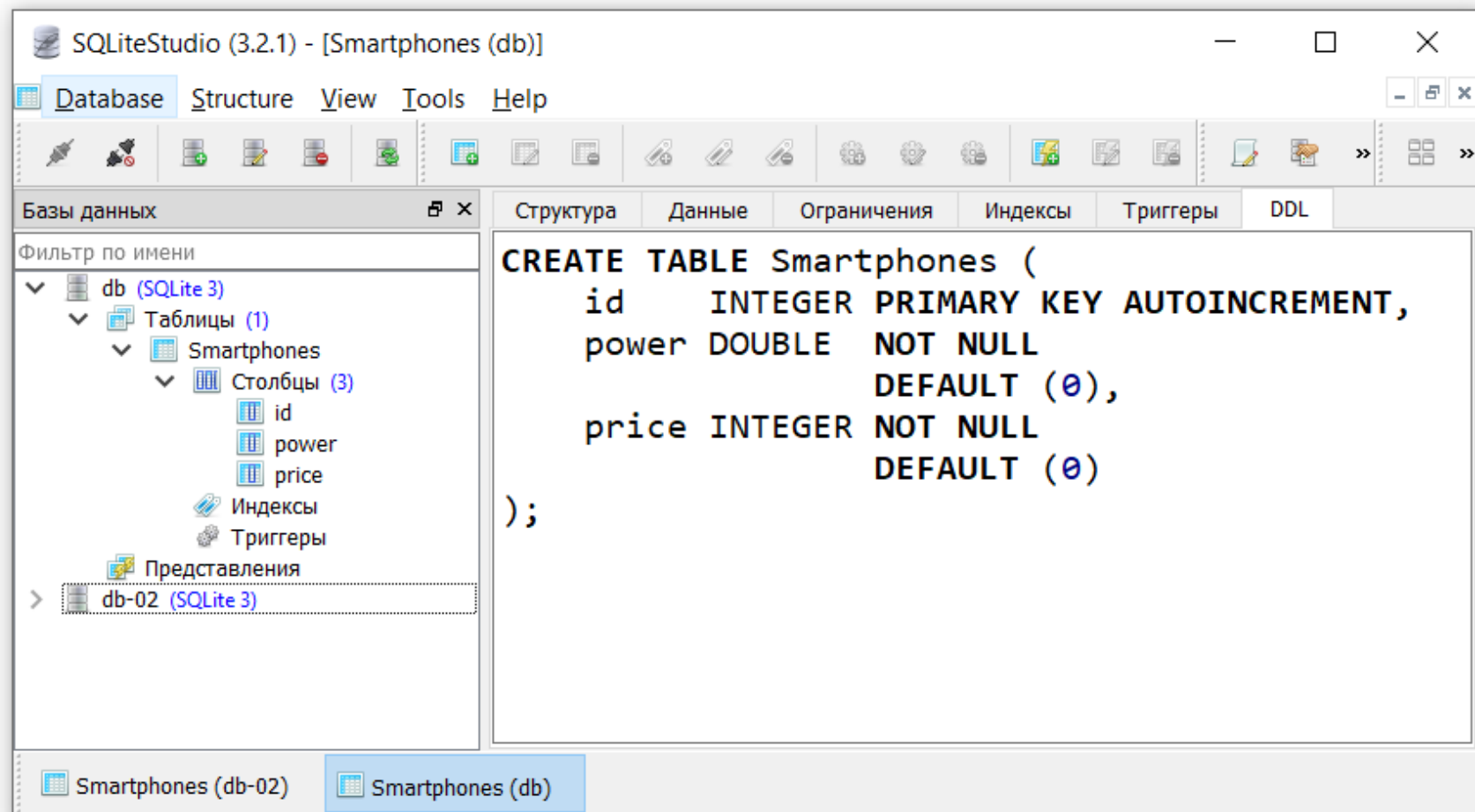
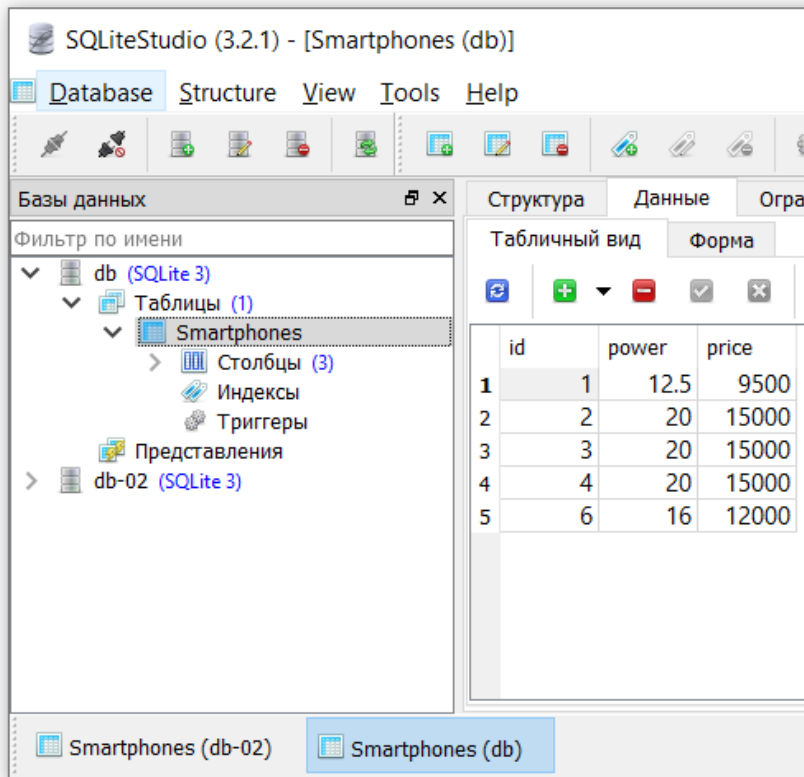
SQLiteStudio-3.2.1.dmg (*portable*)

InstallSQLiteStudio-3.2.1.dmg (*installer*)

<https://github.com/pawelsalawa/sqlitestudio/releases>



Интерфейс SQLiteStudio



Редактор SQL

The screenshot displays the SQL Editor application interface. On the left, a tree view shows the database structure under 'Базы данных'. The selected database is 'db (SQLite 3)', which contains a table 'Smartphones' with columns 'id', 'power', and 'price'. Below it, another database 'db-02 (SQLite 3)' also contains a 'Smartphones' table with the same columns.

The main area is divided into two tabs: 'Запрос' (Query) and 'История' (History). The 'Запрос' tab is active, showing a list of SQL queries. The third query is selected and highlighted in light blue:

```
3 SELECT power, price FROM Smartphones;
```

Below the query editor, there are two tabs: 'Табличный вид' (Table view) and 'Форма' (Form). The 'Табличный вид' tab is active, displaying a table with 9 rows and 2 columns: 'power' and 'price'. The data is as follows:

	power	price
5	10	10000
6	21.4	9000
7	16.5	8500
8	14.8	8800
9	17	9000

Below the table view, there is a 'Статус' (Status) section showing a log of executed queries. The log entries are:

- [22:01:11] Запрос выполнен за 0.001 секунд.
- [22:01:55] Запрос выполнен за 0.002 секунд.
- [22:02:39] Запрос выполнен за 0.003 секунд.
- [22:07:28] Запрос выполнен за 0.251 секунд. Затронуто строк: 3
- [22:07:39] Запрос выполнен за 0.001 секунд.

The bottom status bar shows three active windows: 'Smartphones (db-02)', 'Smartphones (db)', and 'Редактор SQL 1'.

Редактор данных

Базы данных

Фильтр по имени

- db (SQLite 3)
 - Таблицы (1)
 - Smartphones
 - Столбцы (3)
 - id
 - power
 - price
 - Индексы
 - Триггеры
 - db-02 (SQLite 3)
 - Таблицы (1)
 - Smartphones
 - Столбцы (3)
 - id
 - power
 - price
 - Индексы
 - Триггеры
 - Представления

Структура | Данные | Ограничения | Индексы | Триггеры | DDL

Имя таблицы: Smartphones ☐ WITHOUT ROWID

	Имя	Тип данных	Первичный ключ	Внешний ключ	Уникальность	Проверка	Не NULL	Сравнение	Значение по умолчанию
1	id	INTEGER							NULL
2	power	DOUBLE							
3	price	INTEGER							

Редактировать ограничение

Первичный ключ

☒ Автоинкремент

☐ Порядок сортировки: ASC

☐ Именованное ограничение:

☐ При конфликте: ROLLBACK

Применить Cancel

Столбец

Имя и тип

Имя столбца: id Тип данных: INTEGER Размер: ,

Ограничения

 - ☒ Первичный ключ Настроить
 - ☐ Внешний ключ Настроить
 - ☐ Уникальность Настроить
 - ☐ Проверка условия Настроить
 - ☐ Не NULL Настроить
 - ☐ Сравнение Настроить
 - ☐ Default Настроить

☐ Расширенный режим OK Cancel

[22:07:28] Запрос выполнен за 0.251 секунд. Затронуты 0 записей.

[22:07:39] Запрос выполнен за 0.001 секунд.

Smartphones (db-02) Smartphones (db) Редактор SQL 1

Примеры запросов

```
SELECT * FROM Smartphones;
```

```
INSERT INTO Smartphones(power, price) VALUES (9, 8500);
```

```
INSERT INTO Smartphones(power, price) VALUES (16.5, 8500), (14.8, 8800), (17.0, 9000);
```

```
DELETE FROM Smartphones WHERE price = 8500;
```

```
DELETE FROM Smartphones;
```

```
/*
```

многострочный комментарий

```
*/
```

```
--
```

однострочный комментарий

Примеры запросов

```
SELECT power, price FROM Smartphones;
```

```
SELECT power, price FROM Smartphones WHERE price < 22000;
```

```
SELECT power, price FROM Smartphones WHERE price < 22000 ORDER BY price DESC;
```

```
SELECT DISTINCT price FROM Smartphones;
```

```
SELECT count(DISTINCT price) FROM Smartphones;
```

```
SELECT power, price FROM Smartphones GROUP BY price;
```

```
SELECT power, SUM(price), COUNT(*) AS amount FROM Smartphones GROUP BY power;
```

Примеры запросов

```
UPDATE Smartphones SET price = 10500 WHERE price = 10000;
```

```
UPDATE Smartphones SET price = 15000 WHERE price > 10000 AND price < 20000;
```

```
UPDATE Smartphones SET price = 15666 WHERE price BETWEEN 10000 AND 20000;
```

```
UPDATE Smartphones SET price = 17000 WHERE power IN (9,10,11);
```

```
UPDATE Smartphones SET price = CASE power  
    WHEN 9 THEN 19000  
    WHEN 8 THEN 18000  
    WHEN 7 THEN 17000  
    ELSE price END;
```


Вариант итогового задания

- 1) считать из CSV-файла данные ([репозиторий](#))
- 2) загрузить их в таблицу в SQLite БД (редактором, программой на C#)
- 3) написать программу выбора Парето-оптимального решения
 - с dll-библиотекой формирования списка оптимальных решений

где можно указать:

**направление оптимизации и
размерность**

ВидеоУроки:

[Оптимизация по Парето. Часть 1.](#)

[Оптимизация по Парето. Часть 2.](#)

Вариант **курсового проекта**:

всё это + WFA (или WPF) + 2D отображение результатов на форме