В прошлых уроках мы создали своё первое серьёзное приложение, позволяющее устанавливать и удалять приложения.

Теперь же мы начнём создавать уже второе приложение — для хранения заметок. И, как и раньше, при создании приложения постепенно будем получать новые навыки и знания, необходимые для каждого разработчика.

Давайте же сформулируем задачу, которую мы решим за ближайшие несколько уроков: создать приложение, которое позволит создавать, хранить, удалять и редактировать заметки. Приложение должно уметь не только сохранять текст (это слишком уж банально), но и прикреплять фотографии.

Ну и, конечно же, всё это будет реализовано с использованием Material Design.

Работа с SQLite

В Android, как и в iOS, есть встроенная в систему компактная база данных — **SQLite**.

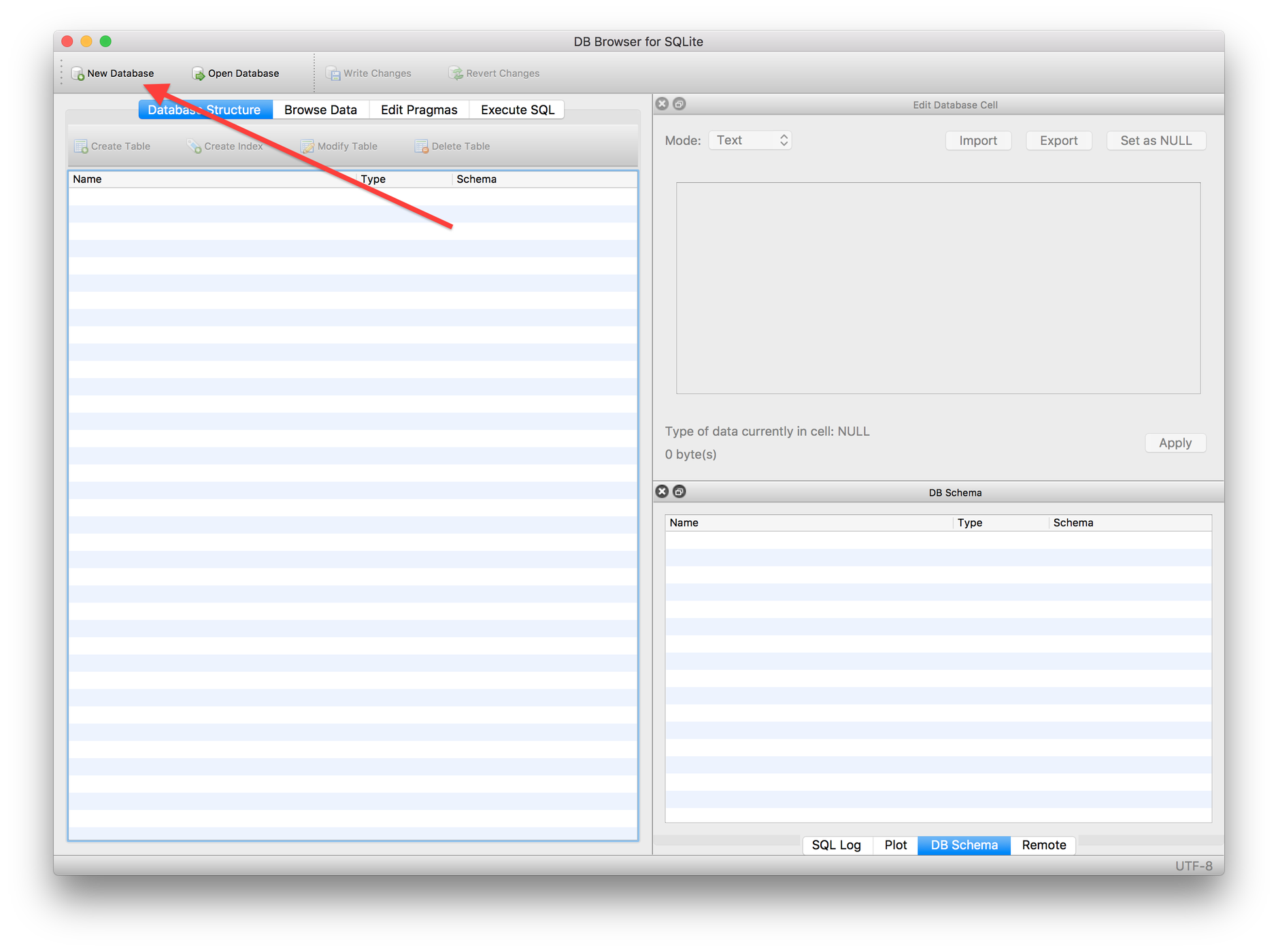
SQLite обладает рядом недостатков — например, в него нельзя писать данные одновременно из нескольких потоков. Однако, все эти недостатки перекрываются огромным (для мобильных устройств) преимуществом: она чрезвычайно компактна.

Вся база данных хранится в одном-единственном файле, и, хоть и работает она относительно медленно (по сравнению со "старшими братьями"; для своих целей она является очень даже шустрой), но при этом занимает очень мало места и способна работать даже на самых слабых устройствах.

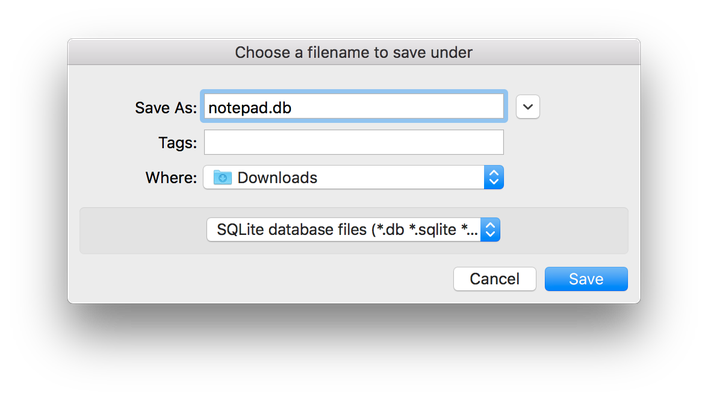
Чтобы понять, как оно работает, давайте попробуем поработать с SQLite на компьютере. Если вы не знакомы с SQL, рекомендую ознакомиться с [этой статьёй](https://ru.wikipedia.org/wiki/SQL) и ссылками в ней.

В практике работы с SQLite нам поможет приложение [SQLite Database Browser](http://sqlitebrowser.org/). Оно полностью бесплатно, работает на macOS, Windows и Linux, и исходный код доступен на GitHub. Скачайте и установите его.

После установки запустите приложение и нажмите на кнопку **New Database**:

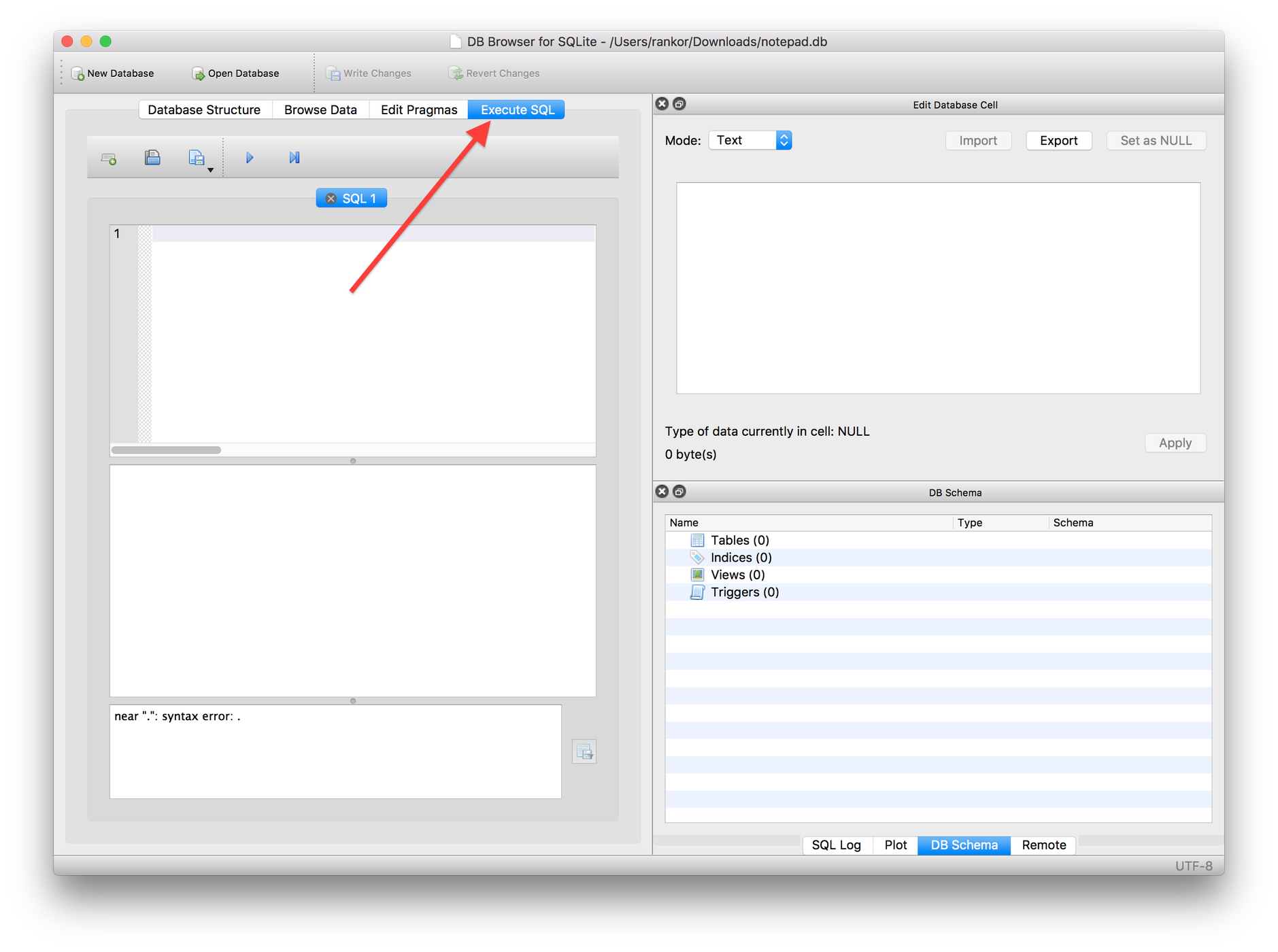
[](https://skillberg.com/media/uploads/2017/11/17/6f4bf7d91afd43f49cf23d280923017d.png)Кнопка "New Database"

Введите имя и расположения файла, в котором будет создана БД и нажмите на кнопку **Save**:

[](https://skillberg.com/media/uploads/2017/11/17/bdca12767bab469e8dee8c2d3d593b32.png)Выбор файла для сохранения БД

После этого появится диалог, предлагающий создать таблицу. Закройте его — мы будем создавать её вручную.

Выберите вкладку **Execute SQL**:

[](https://skillberg.com/media/uploads/2017/11/17/5750920f3b3143afa0286a945efaa387.png)Вкладка "Execute SQL"

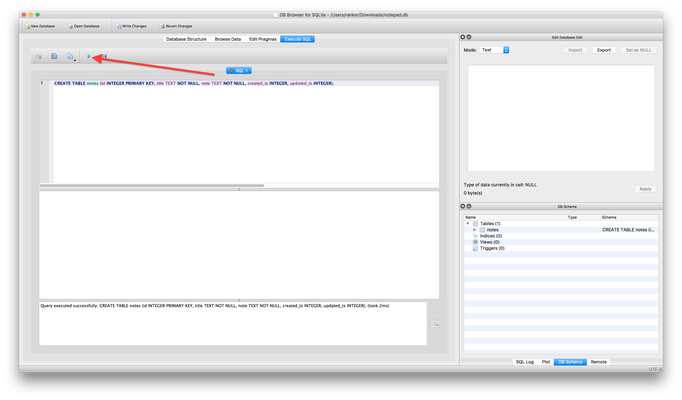
Для начала нам нужна одна таблица, в которой мы будем хранить заметки. В таблице будут такие столбцы:

* id — integer, primary key. Идентификатор заметки.
* title — text, заголовок заметки.
* note — text, текст заметки.
* created\_ts — integer, дата создания заметки в формате [unix time](https://ru.wikipedia.org/wiki/UNIX-%D0%B2%D1%80%D0%B5%D0%BC%D1%8F).
* updated\_ts — integer, дата последнего изменения заметки в формате unix time.

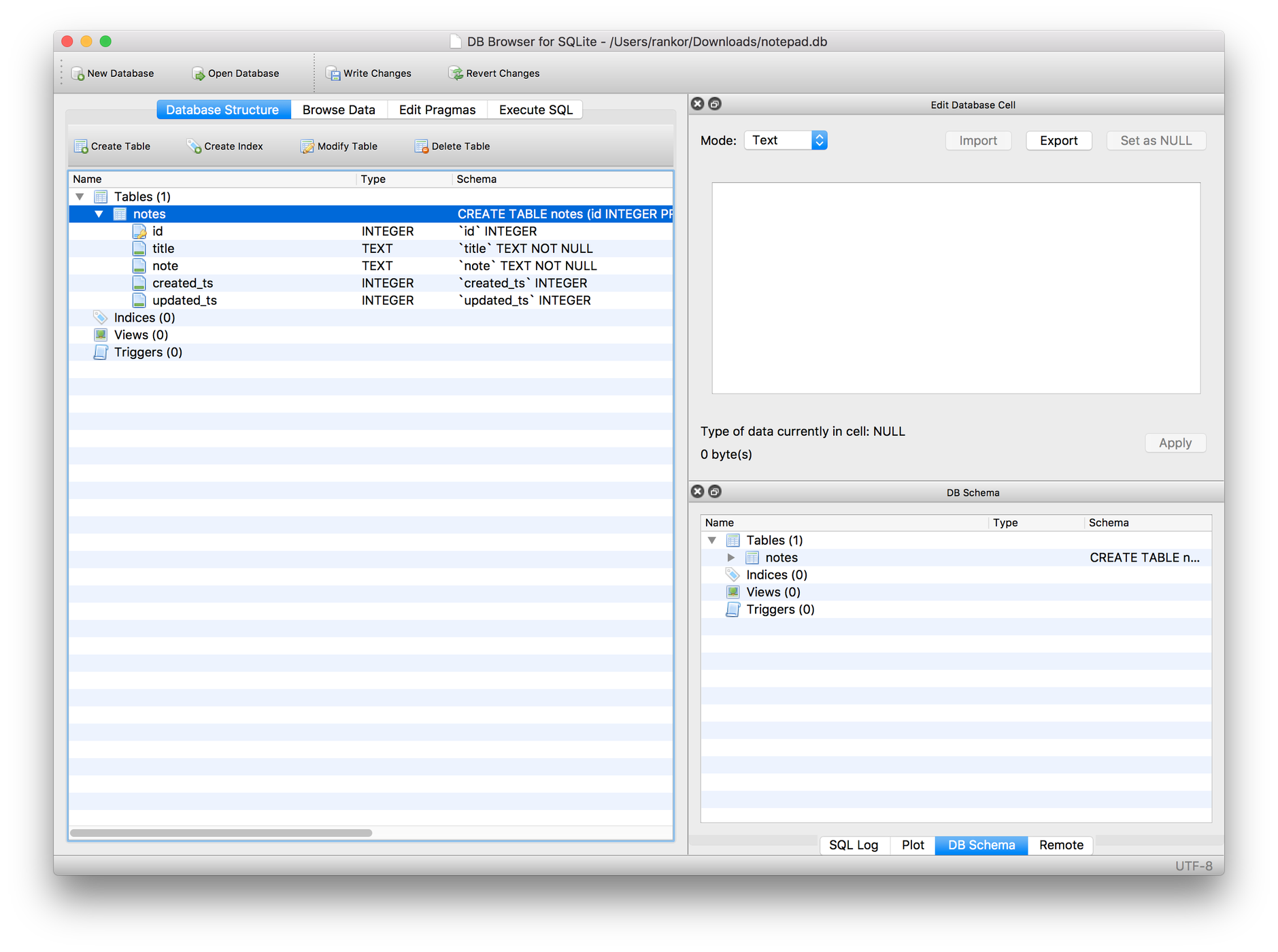
SQL-запрос для создания таблицы будет таким:

CREATE TABLE notes (id INTEGER PRIMARY KEY, title TEXT NOT NULL, note TEXT NOT NULL, created\_ts INTEGER, updated\_ts INTEGER);

Введите его в поле для запроса и нажмите на кнопку **Execute SQL**. Результат выполнения появится в окне ниже:

[](https://skillberg.com/media/uploads/2017/11/17/b43662f9ec5346db8e856b3954a0e2a6.png)Результат выполнения SQL

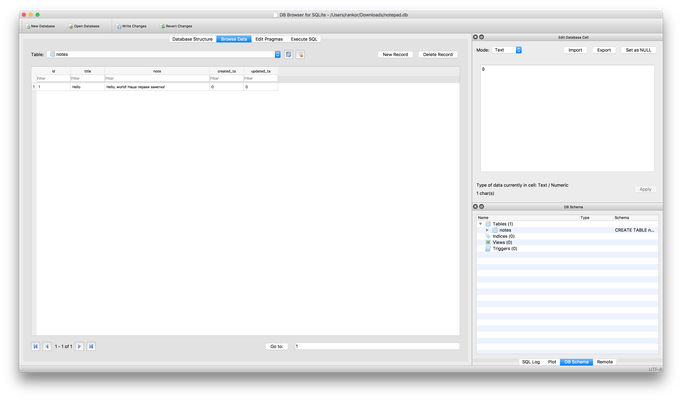
Теперь вернемся на вкладку **Database Structure** и посмотрим, что получилось:

[](https://skillberg.com/media/uploads/2017/11/17/f335af82d0924b2ebcc23445cc88a03b.png)Структура БД SQLite

Отлично. Теперь вставим первую заметку в нашу таблицу:

INSERT INTO notes (title, note, created\_ts, updated\_ts) VALUES ('Hello', 'Hello, world! Наша первая заметка!', 0, 0);

В качестве времени создания и обновления для простоты передадим 0, то есть начало эпохи UNIX. Выполните этот запрос и перейдите на вкладку **Browse Data**:

[](https://skillberg.com/media/uploads/2017/11/17/8a321bc931d94fc690a0a57776cf39ca.png)Данные в БД SQLite

Как видите, данные вставились успешно. Добавьте ещё одну заметку:

INSERT INTO notes (title, note, created\_ts, updated\_ts) VALUES ('Вторая заметка', 'А это уже вторая заметка', 123, 456);

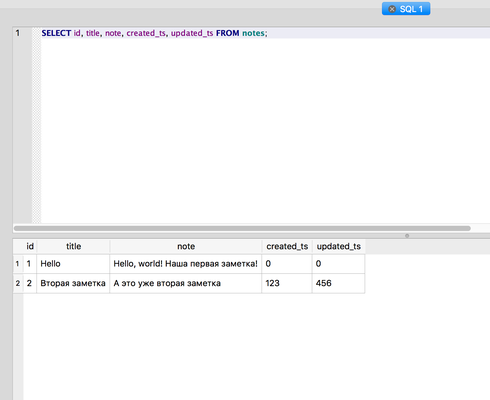
У неё мы изменили время создания и редактирования.

Зачем? Чтобы показать работу сортировки в SQLite. Мы будем показывать первыми те заметки, которые были изменены последними.

Если вы выполните следующий запрос:

SELECT id, title, note, created\_ts, updated\_ts FROM notes;

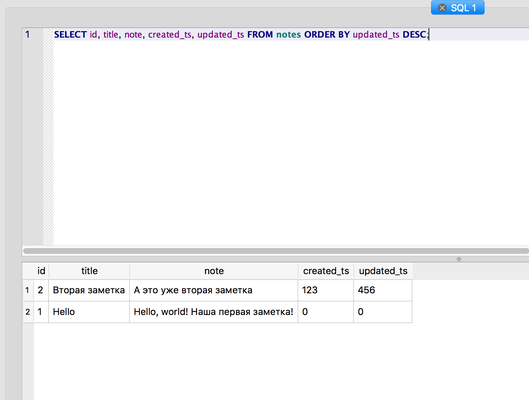
То получите вот такой результат:

[](https://skillberg.com/media/uploads/2017/11/17/2a77c3aa739a413c82e32255ef1b796d.png)Результат выполнения SELECT

То есть первой показывается первая заметка. А теперь выполните вот такой запрос:

SELECT id, title, note, created\_ts, updated\_ts FROM notes ORDER BY updated\_ts DESC;

В нём мы говорим базе данных, что нужно отсортировать данные по полю updated\_ts, при этом сортировку нужно выполнить по убыванию. Результатом будет следующий набор данных:

[](https://skillberg.com/media/uploads/2017/11/17/4e72ddfe412040b18728c77c636c25e6.png)Данные в БД SQLite с сортировкой

Отлично.

Теперь давайте попробуем обновить заметку:

UPDATE notes SET title = 'Первая заметка', note = 'Всё та же первая заметка, но обновленная', updated\_ts = 890 WHERE id = 1;

Главное — не забыть WHERE, иначе обновятся **все** строки в таблице.

Ну и удалим заметку:

DELETE FROM notes WHERE id = 1;

Последняя вещь, о которой нужно узнать — это [индексы](https://ru.wikipedia.org/wiki/%D0%98%D0%BD%D0%B4%D0%B5%D0%BA%D1%81_(%D0%B1%D0%B0%D0%B7%D1%8B_%D0%B4%D0%B0%D0%BD%D0%BD%D1%8B%D1%85)). Если у пользователя будет много заметок, то сортировка может занимать достаточно продолжительное время, в результате, список заметок будет отображаться с лагами. Поэтому нам нужно создать индекс по столбцу updated\_ts:

CREATE INDEX updated\_ts\_index ON notes (updated\_ts);

Вот и всё! Примерно то же самое мы будем делать в нашем приложении.

SQLite в Android

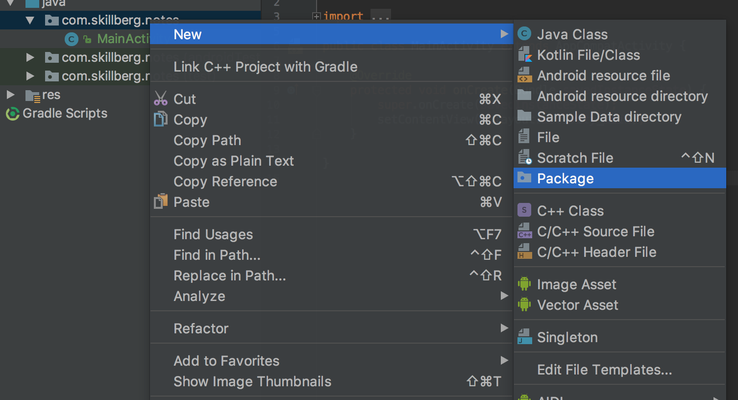
Работа с SQLite в Android отличается от того, что мы делали ранее всего парой аспектов:

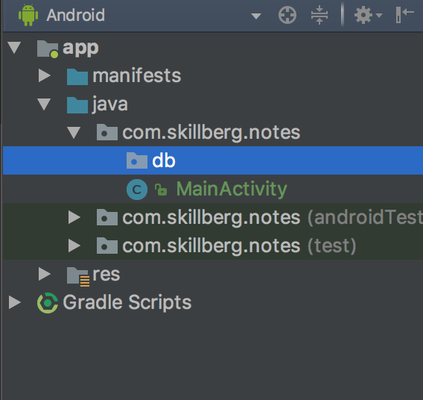
* Вам почти никогда не придётся работать с "сырыми" SQL-запросами — для вставки/получения/удаления записей есть удобные методы.
* Стандартное для большинства движков БД имя столбца id в Android по какой-то неведомой причине принято заменять на \_id. Лучше следовать этому стандарту, иначе некоторые вещи могут сломаться.

Базы данных хранятся в приватных директориях приложений, то есть они недоступны для других приложений.

Создайте новый проект, как мы делали это ранее. В прошлом проекте мы хранили все файлы в одном пакете. В этом мы более аккуратно подойдём к организации. Начнём, собственно, с БД.

Внутри основного пакета создайте пакет db:

[](https://skillberg.com/media/uploads/2017/11/17/4762e6acbd344977ae541c86b0ba5733.png)Создание нового пакета

[](https://skillberg.com/media/uploads/2017/11/17/c950ba3782164f7ebbc87012b94c206b.png)Пакет "db"

Контракт БД

В пакете db, который мы создали ранее, создайте класс NotesContract:

public final class NotesContract {

private NotesContract() {

}

public static abstract class Notes implements BaseColumns {

}

}

Этот класс облегчит работу с БД, а так же будет выступать в качестве "документации" структуры нашей БД для тех, кто возможно будет смотреть наши исходники. В нём мы будем описывать таблицы, которые будут присутствовать в БД.

Класс определён как final, чтобы от него нельзя было наследоваться, а так имеет пустой приватный конструктор — чтобы никто случайно не создал инстанс класса.

Класс Notes будет содержать описание таблицы с заметками: имя таблицы и столбцы. Он реализует интерфейс BaseColumns, в котором определены константы \_ID и \_COUNT, необходимые для корректной работы некоторых вещей при взаимодействии с БД.

Давайте добавим название таблицы и столбцы в класс Notes:

public static abstract class Notes implements BaseColumns {

public static final String TABLE\_NAME = "notes";

public static final String COLUMN\_TITLE = "title";

public static final String COLUMN\_NOTE = "note";

public static final String COLUMN\_CREATED\_TS = "created\_ts";

public static final String COLUMN\_UPDATED\_TS = "updated\_ts";

}

Такой подход считается хорошей практикой, поскольку он не только документирует структуру таблиц, но и позволяет не допускать ошибок (например, опечаток) при взаимодействии с БД.

Так же давайте добавим в Notes строки, в которых будут содержаться запросы, создающие таблицу и индекс:

public static final String CREATE\_TABLE = String.format("CREATE TABLE %s " +

"(%s INTEGER PRIMARY KEY, " +

"%s TEXT NOT NULL, " +

"%s TEXT NOT NULL, " +

"%s INTEGER NOT NULL, " +

"%s INTEGER NOT NULL);",

TABLE\_NAME,

\_ID,

COLUMN\_TITLE,

COLUMN\_NOTE,

COLUMN\_CREATED\_TS,

COLUMN\_UPDATED\_TS);

public static final String CREATE\_UPDATED\_TS\_INDEX = String.format("CREATE INDEX updated\_ts\_index " +

"ON %s (%s);",

TABLE\_NAME,

COLUMN\_UPDATED\_TS);

Ну и в классе NotesContract создадим массив, в котором будут перечислены запросы, требующиеся для создания БД, а также имя и версию БД:

public final class NotesContract {

public static final String DB\_NAME = "notes.db";

public static final int DB\_VERSION = 1;

public static final String[] CREATE\_DATABASE\_QUERIES = {

Notes.CREATE\_TABLE,

Notes.CREATE\_UPDATED\_TS\_INDEX

};

// ....

SQLiteOpenHelper

Для того, чтобы облегчить создание и обновление БД, в Android SDK существует класс SQLiteOpenHelper. Давайте создадим его подкласс в пакете db:

public class NotesDbHelper extends SQLiteOpenHelper {

public NotesDbHelper(Context context) {

super(context, NotesContract.DB\_NAME, null, NotesContract.DB\_VERSION);

}

@Override

public void onCreate(SQLiteDatabase sqLiteDatabase) {

for (String query : NotesContract.CREATE\_DATABASE\_QUERIES) {

sqLiteDatabase.execSQL(query);

}

}

@Override

public void onUpgrade(SQLiteDatabase sqLiteDatabase, int i, int i1) {

}

}

В нём мы обязательно должны переопределить следующие методы:

* Конструктор. Первый параметр — контекст, второй — имя БД, третий — фабрика Cursor (просто передаём null), четвёртый — версия БД.
* onCreate() — вызывается, когда БД создаётся. В этом методе мы должны выполнить все запросы, необходимые для создания таблиц.
* onUpgrade() — вызывается, когда система обнаруживает, что версия БД поменялась. В этом методе мы должны выполнить все запросы, необходимые для обновления структуры БД.

На этом пока что всё. В следующем уроке мы начнём создавать и получать данные из нашей базы данных, используя ContentProvider.

ССЫЛКИ

[SQLITEBROWSER](http://sqlitebrowser.org/)

ПРИМЕРЫ КОДА

[ИСХОДНЫЙ КОД УРОКА](https://github.com/Skillberg/NotesAndroid/tree/lesson-20)