Создание SQLite базы данных в DB Browser for SQLite

В статье говорится, как создавать и подготавливать базы данных заранее

Базы данных [SQLite](https://ru.wikipedia.org/wiki/SQLite) очень удобны для работы с небольшими объемами данных как в Windows приложениях, так и в Android приложениях. В статье говорится, как создавать и подготавливать базы данных заранее.

Содержание

* [Установка DB Browser for SQLite](http://blog.harrix.org/article/5154#h2_0)
* [Создание базы данных](http://blog.harrix.org/article/5154#h2_1)

Особенно SQLite удобен тем, что не нужно поднимать сервер СУБД перед использованием в приложениях.

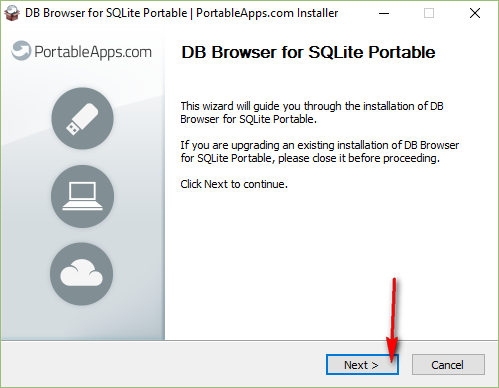
[Установка DB Browser for SQLite](http://blog.harrix.org/article/5154#a_menu)

Итак, скачиваем портабельную версию **DB Browser for SQLite** (**SQLliteBrowser**): <http://sqlitebrowser.org/>

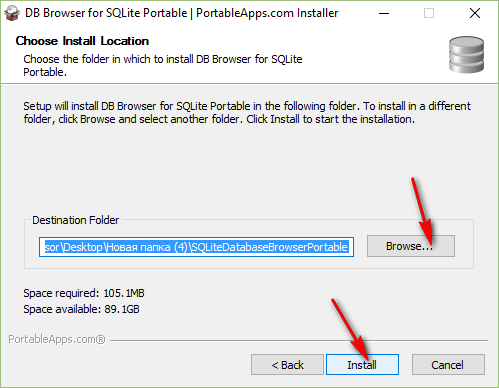
[](http://blog.harrix.org/wp-content/uploads/2016/03/2016-03-21_111650.png)

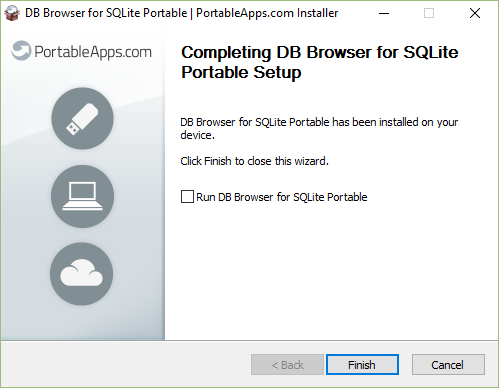
У меня скачался файл **SQLiteDatabaseBrowserPortable\_3.8.0\_English.paf.exe**.

Запускаем его.

[](http://blog.harrix.org/wp-content/uploads/2016/03/2016-03-21_114748.png)

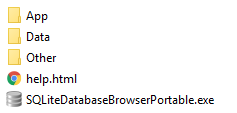
Указываем путь, куда распаковываем портабельную версию.

[](http://blog.harrix.org/wp-content/uploads/2016/03/2016-03-21_114800.png)

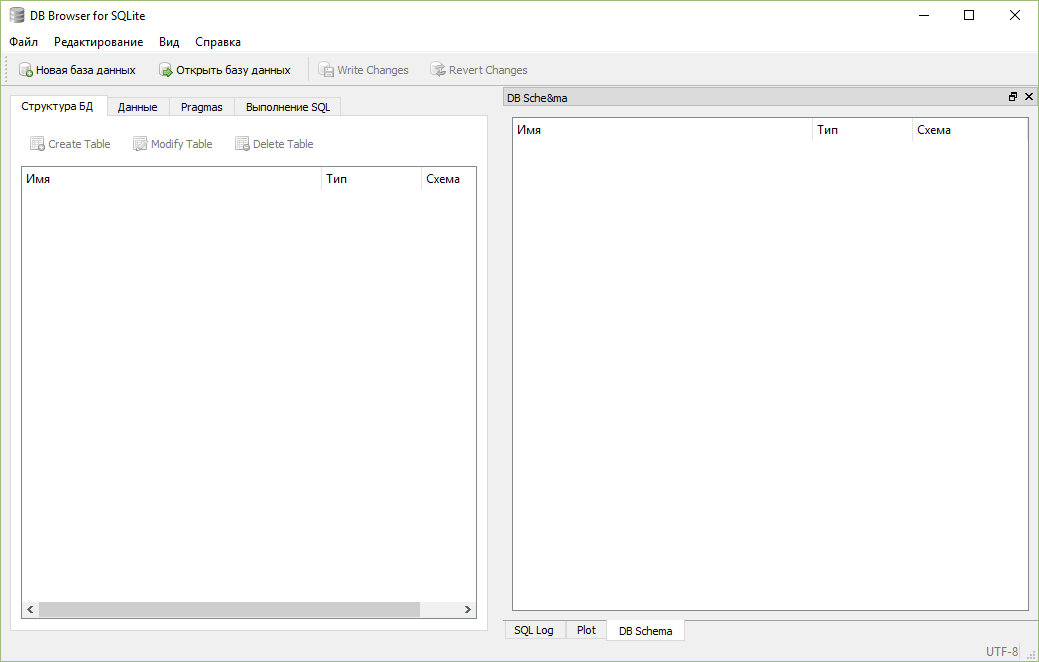
[](http://blog.harrix.org/wp-content/uploads/2016/03/2016-03-21_114825.png)

В итоге, если всё делать по умолчанию, то в папке с нашим установочным пакетом появится папка с портабельной программой.

[2016-03-21_114836](http://blog.harrix.org/wp-content/uploads/2016/03/2016-03-21_114836.png)

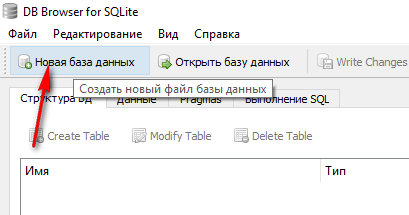
[](http://blog.harrix.org/wp-content/uploads/2016/03/2016-03-21_114845.png)

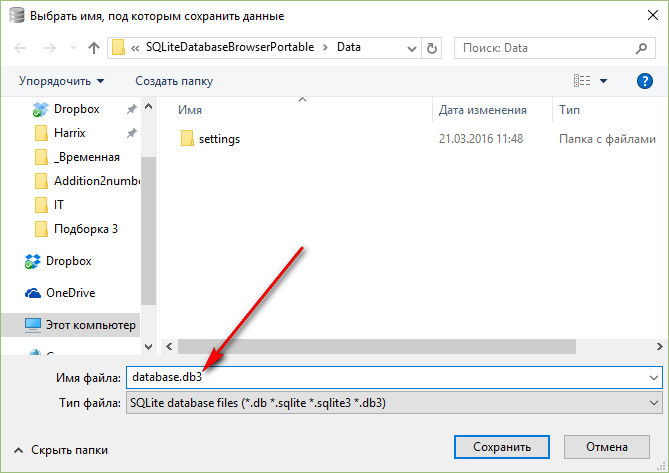
Запускаем нашу программу.

[](http://blog.harrix.org/wp-content/uploads/2016/03/2016-03-21_114853.png)

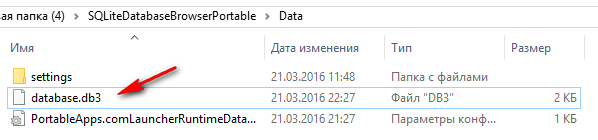
[Создание базы данных](http://blog.harrix.org/article/5154#a_menu)

Создадим простую базу данных **database** с одной таблицей и тремя столбцами.

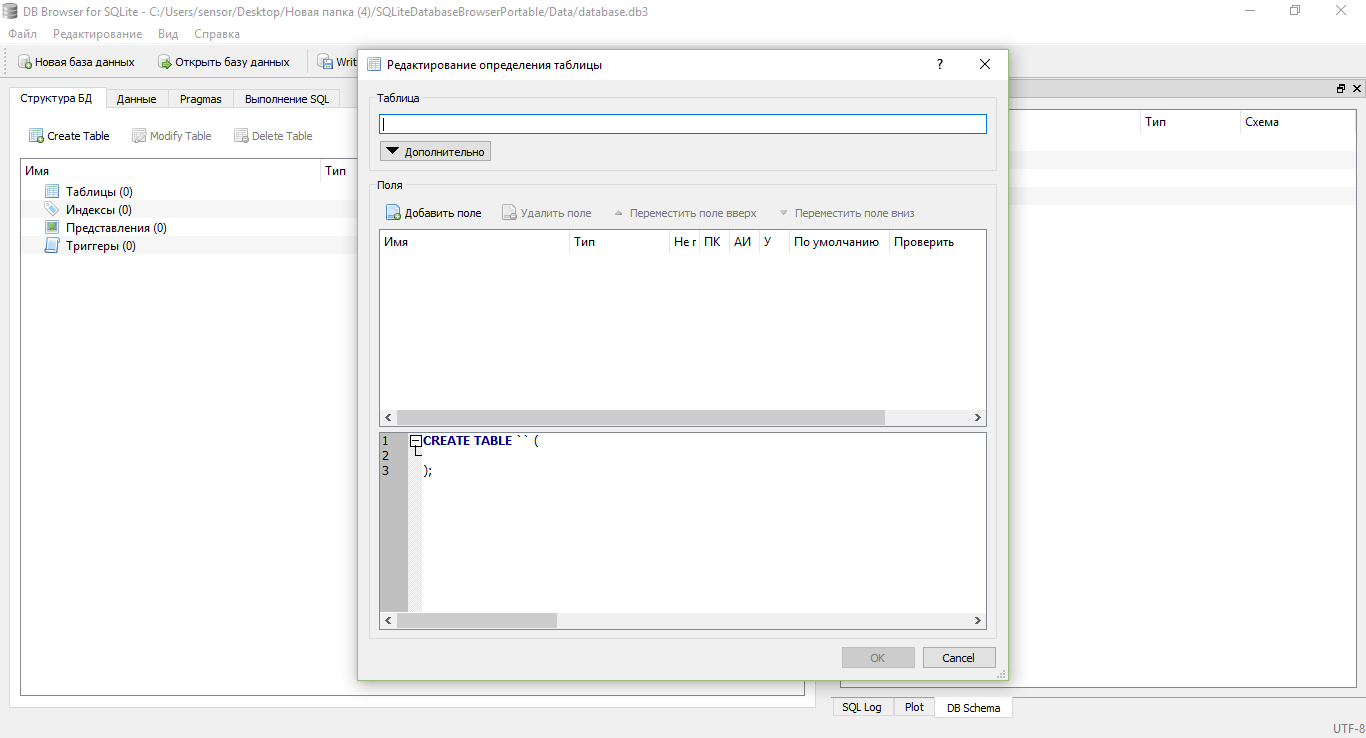
[](http://blog.harrix.org/wp-content/uploads/2016/03/2016-03-21_221923.png)

[](http://blog.harrix.org/wp-content/uploads/2016/03/2016-03-21_222650.png)

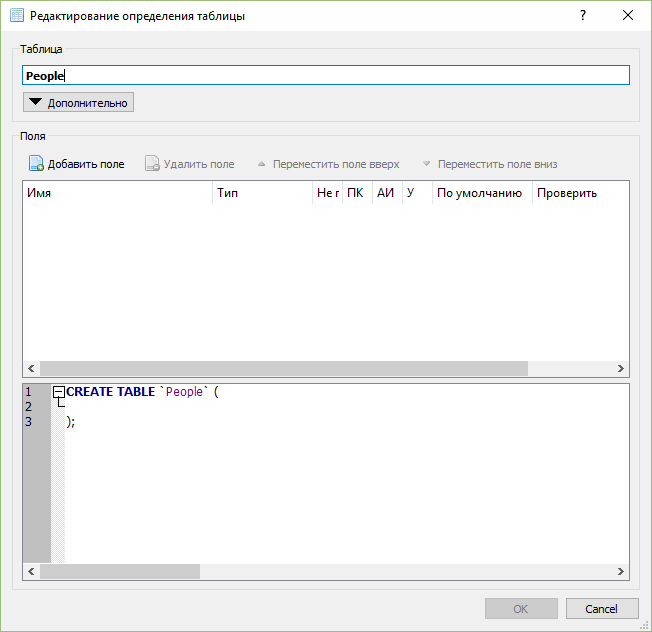
В папке **Data** появится файл вашей базы данных.

[](http://blog.harrix.org/wp-content/uploads/2016/03/2016-03-21_225315.png)

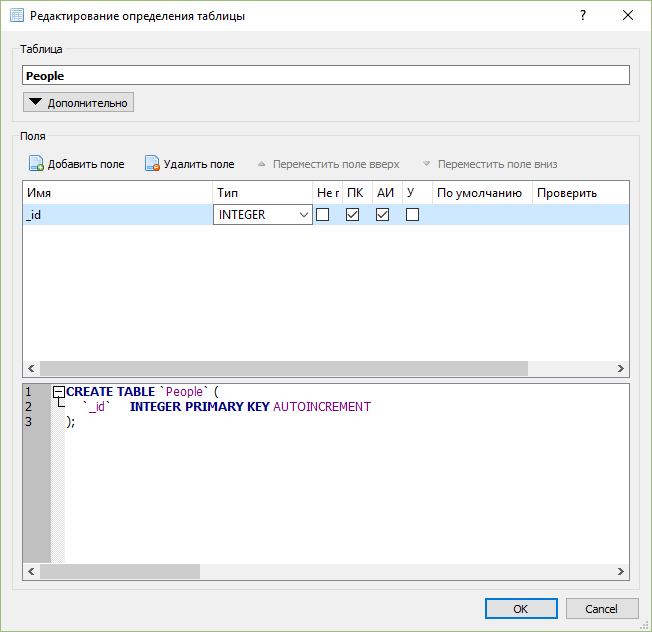
Появится окно создания таблицы в программе.

[](http://blog.harrix.org/wp-content/uploads/2016/03/2016-03-21_222707.png)

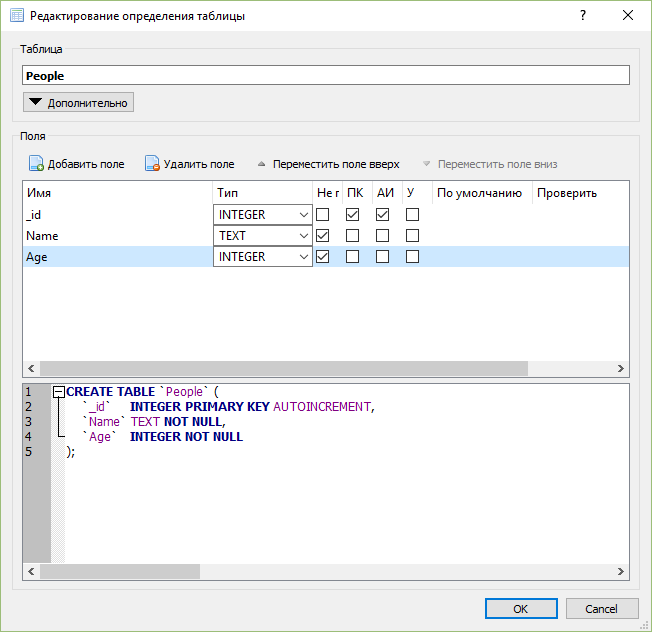
Создадим таблицу **People**.

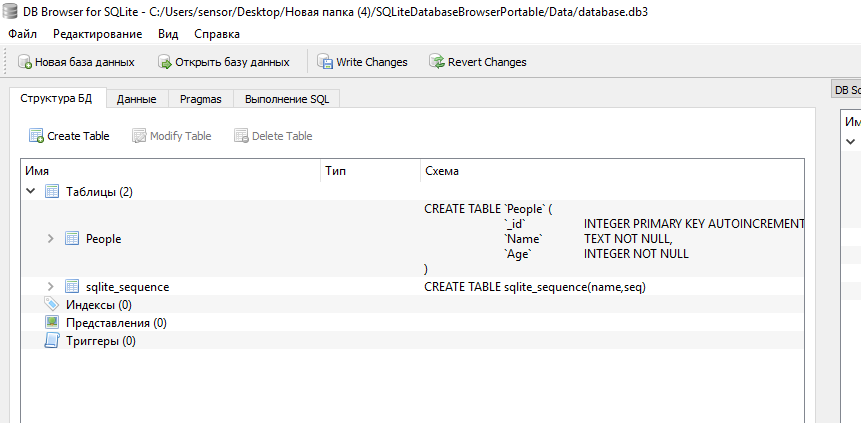
[](http://blog.harrix.org/wp-content/uploads/2016/03/2016-03-21_232530.png)

Ввиду того, что вдруг база данных будет использоваться в Android, то первым полем задавайте индентификатор по имени **\_id**.

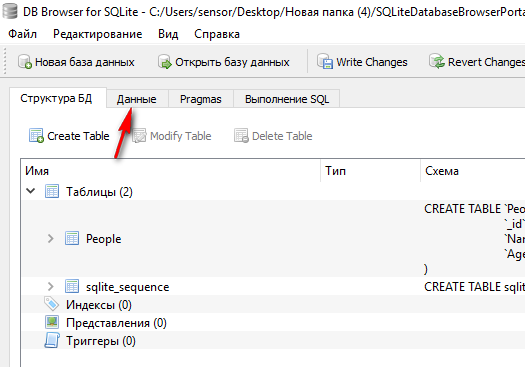
[](http://blog.harrix.org/wp-content/uploads/2016/03/2016-03-21_233230.png)

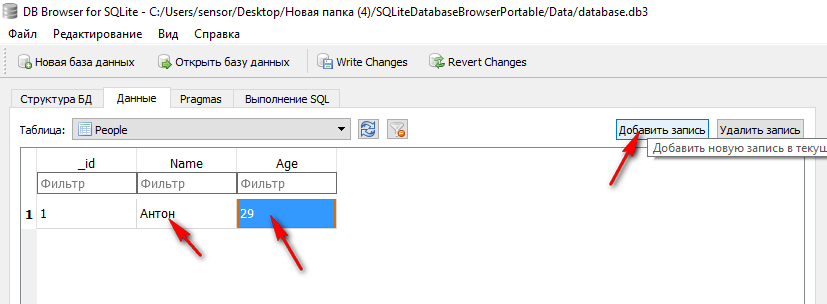
Добавим, например, поля **Name** и **Age**.

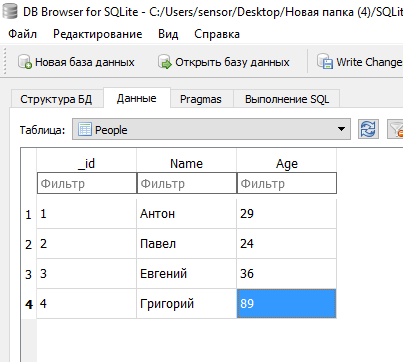
[](http://blog.harrix.org/wp-content/uploads/2016/03/2016-03-21_235329.png)

[](http://blog.harrix.org/wp-content/uploads/2016/03/2016-03-21_235340.png)

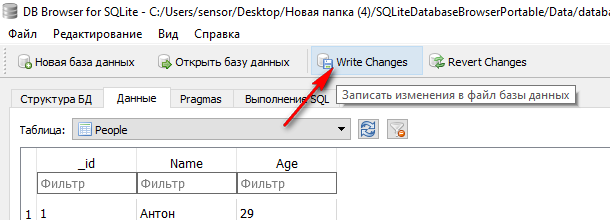
Во вкладке Данные можем вносить данные.

[](http://blog.harrix.org/wp-content/uploads/2016/03/2016-03-21_235444.png)

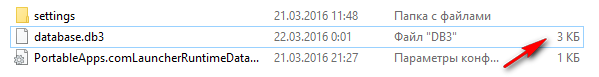
[](http://blog.harrix.org/wp-content/uploads/2016/03/2016-03-22_000016.png)

[](http://blog.harrix.org/wp-content/uploads/2016/03/2016-03-22_000103.png)

Сохраним изменения.

[](http://blog.harrix.org/wp-content/uploads/2016/03/2016-03-22_000113.png)

У вас есть готовая база данных SQLite? которую можете использовать где вам нужно.

[](http://blog.harrix.org/wp-content/uploads/2016/03/2016-03-22_000129.png)

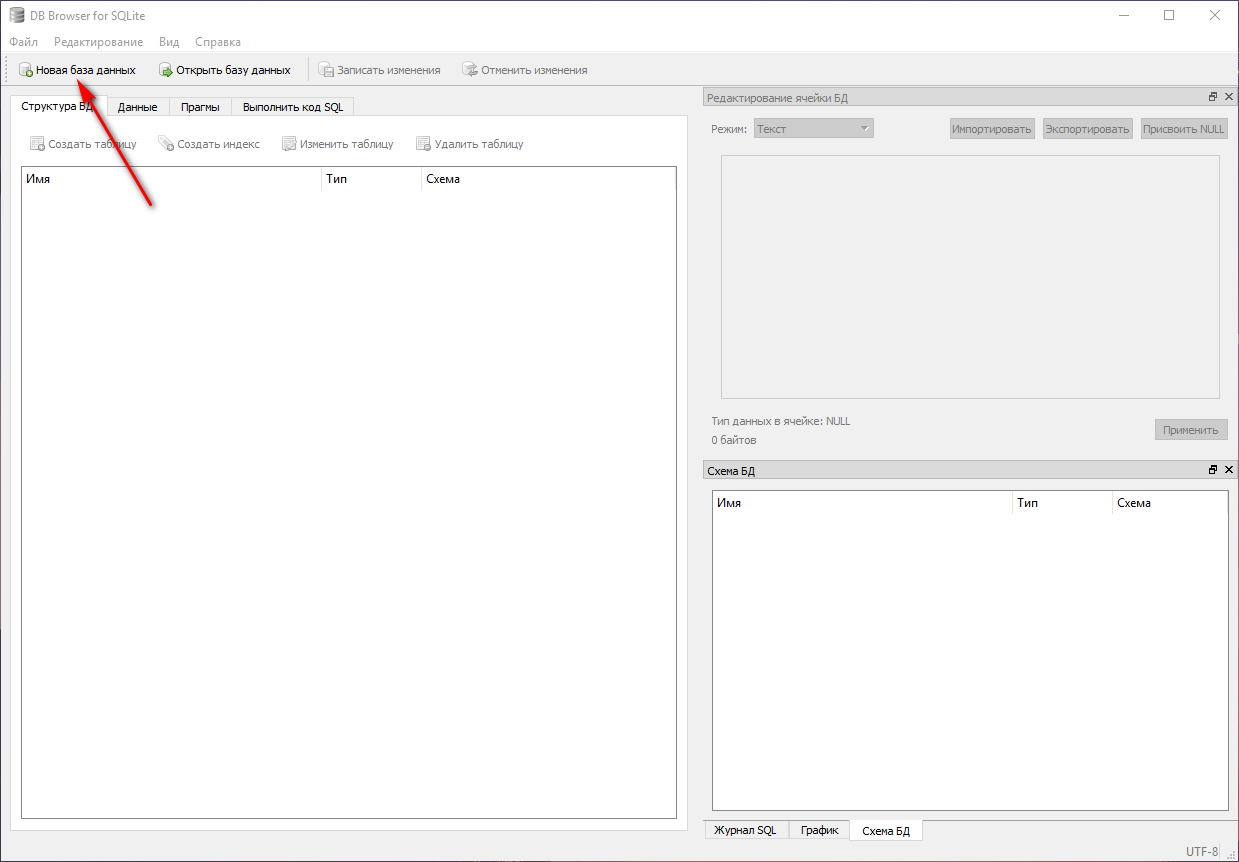
Внешние ключи в DB Browser for SQLite

SQLite поддерживает внешние ключи

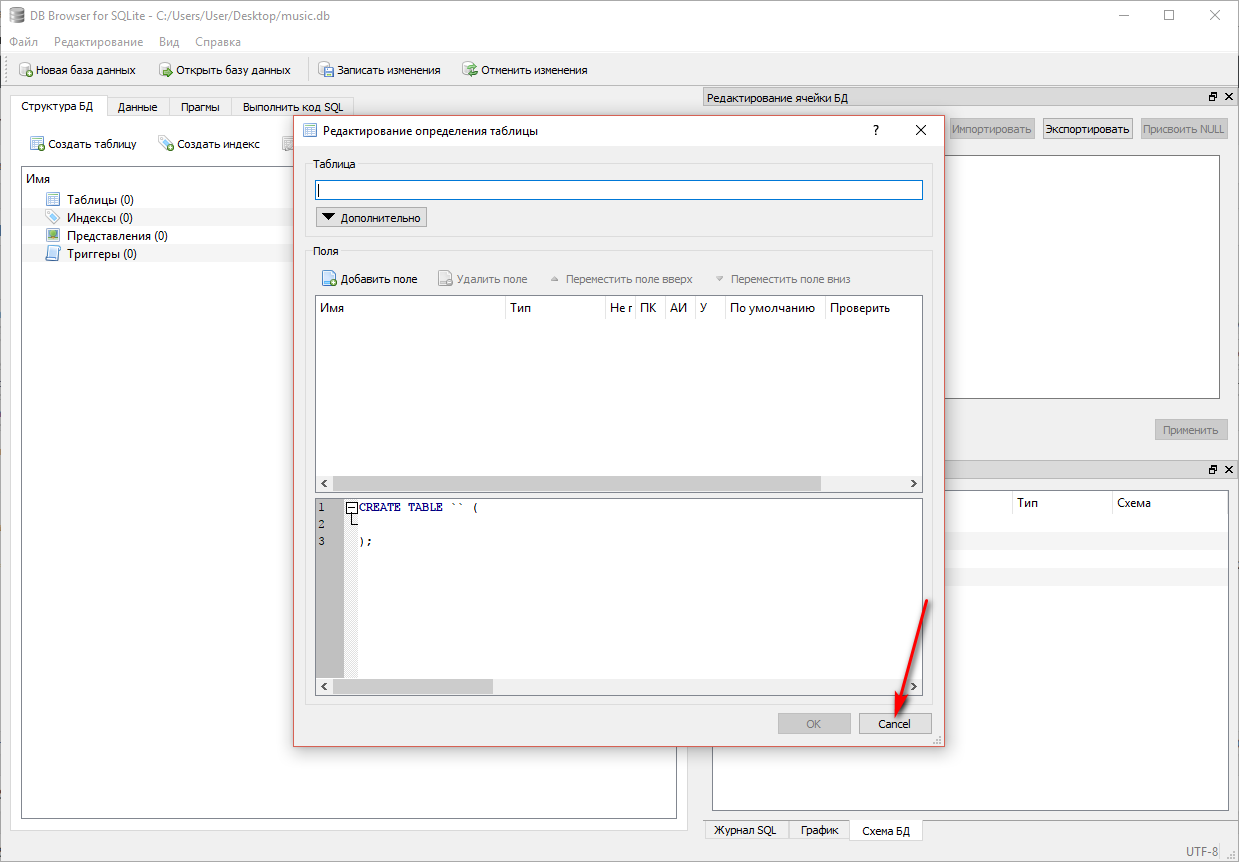
Как включить поддержку внешних ключей в популярной программе по редактированию SQLite баз данных.

Откроем программу **DB Browser for SQLite**.

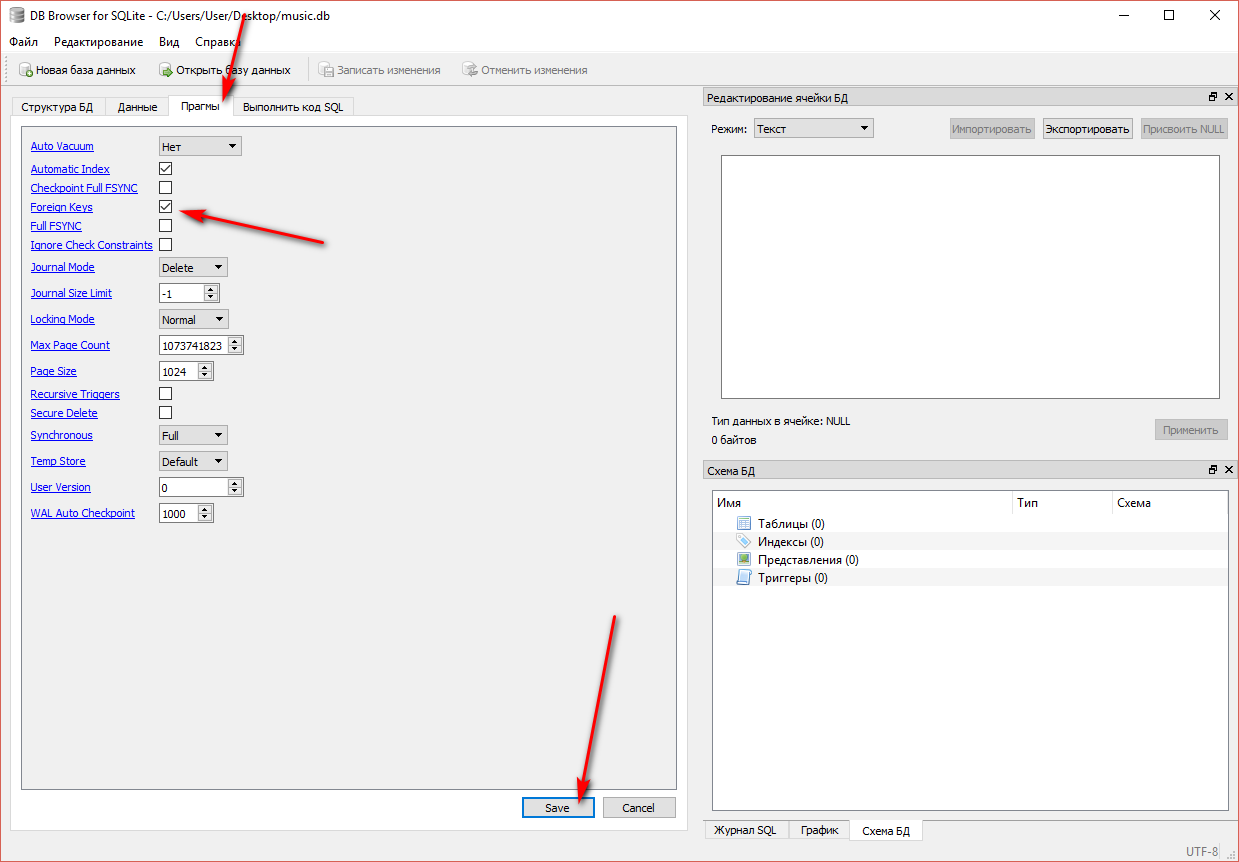
Создадим новую базу данных, например, **music.db**.

[](http://blog.harrix.org/wp-content/uploads/2017/06/2017-06-10_174012.png)

Не будем пока создавать таблицы.

[](http://blog.harrix.org/wp-content/uploads/2017/06/2017-06-10_174126.png)

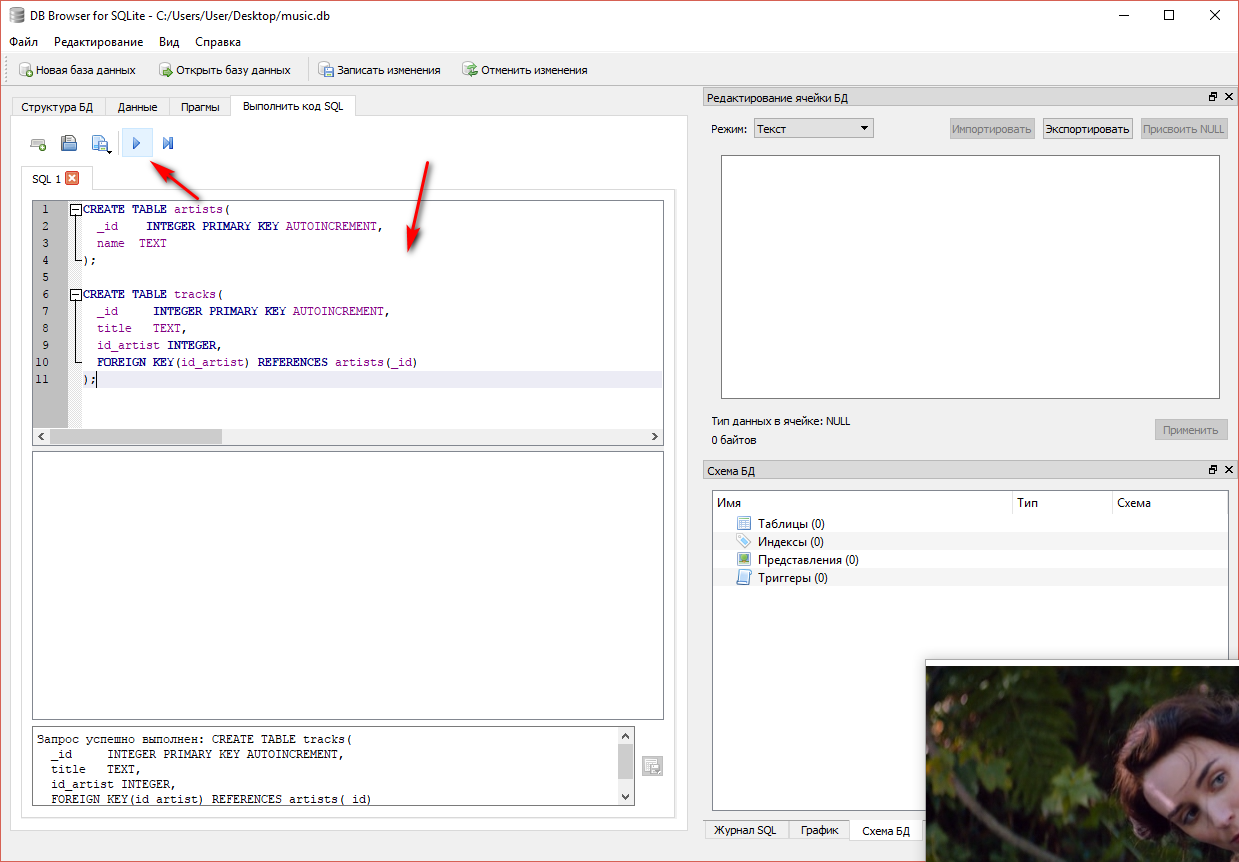
Включим поддержку внешних ключей. Не забудьте нажать на кнопку Save.

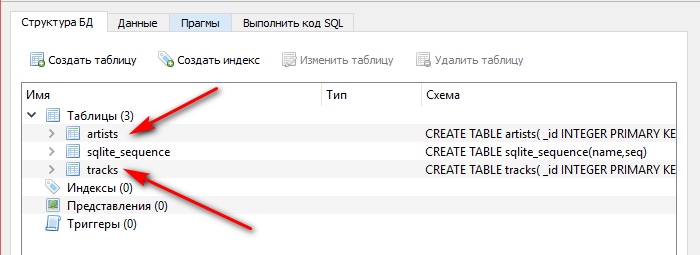
[](http://blog.harrix.org/wp-content/uploads/2017/06/2017-06-10_174218.png)

Теперь покажем, как работают внешние ключи.

Создадим две таблицы **artists** (музыканты) и tracks (песни). Конструктор не будем использовать. Во второй таблице будем хранить номер музыканта как внешний ключ.

|  |
| --- |
| CREATE TABLE artists(    \_id    INTEGER PRIMARY KEY AUTOINCREMENT,    name  TEXT  );    CREATE TABLE tracks(    \_id     INTEGER PRIMARY KEY AUTOINCREMENT,    title   TEXT,    id\_artist INTEGER,  FOREIGN KEY(id\_artist) REFERENCES artists(\_id)  ); |

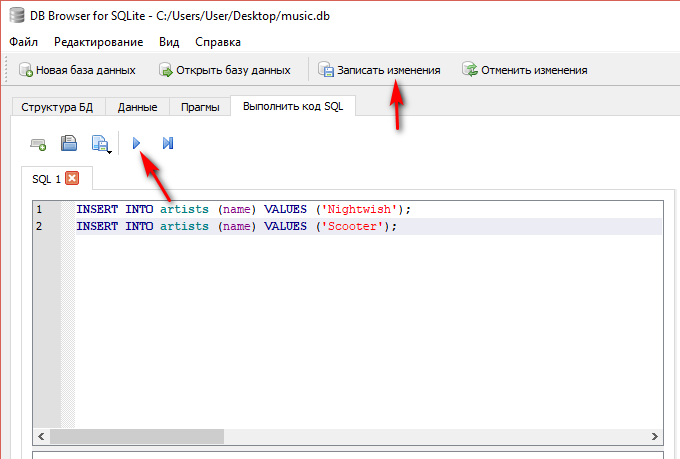
[](http://blog.harrix.org/wp-content/uploads/2017/06/2017-06-10_175335.png)

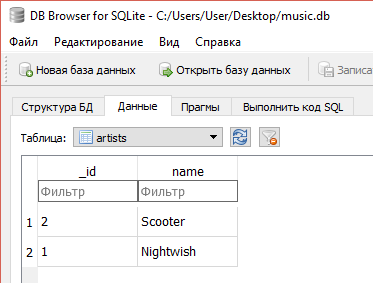
[](http://blog.harrix.org/wp-content/uploads/2017/06/2017-06-10_175402-1.png)

Опять же без конструктора закинем две записи о двух исполнителях.



|  |  |
| --- | --- |
| 1  2 | INSERT INTO artists (name) VALUES ('Nightwish');  INSERT INTO artists (name) VALUES ('Scooter'); |

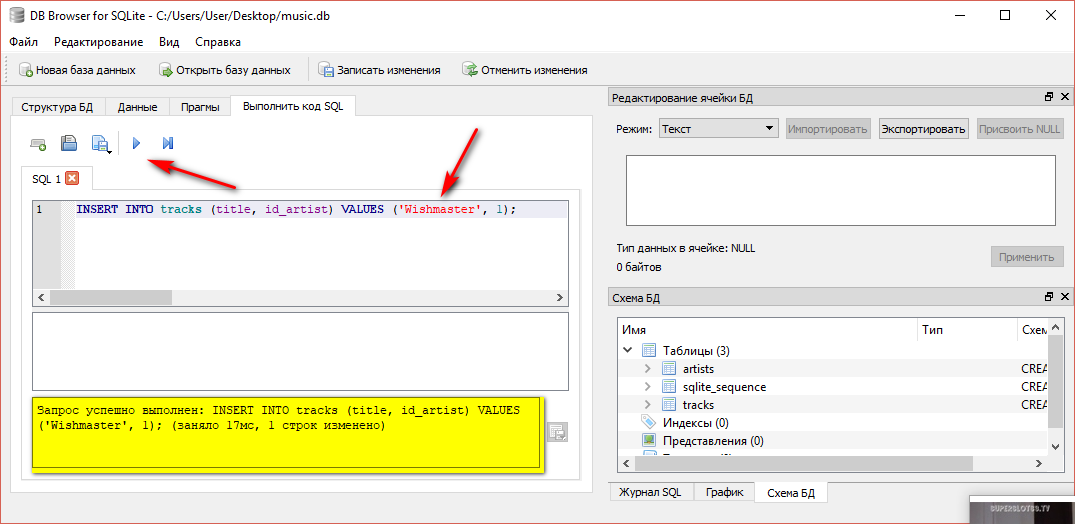
[](http://blog.harrix.org/wp-content/uploads/2017/06/2017-06-10_175814.png)

[](http://blog.harrix.org/wp-content/uploads/2017/06/2017-06-10_175936.png)

Попробуем добавить записи к таблице песню с правильным номером исполнителя.

|  |  |
| --- | --- |
| 1 | INSERT INTO tracks (title, id\_artist) VALUES ('Wishmaster', 1); |

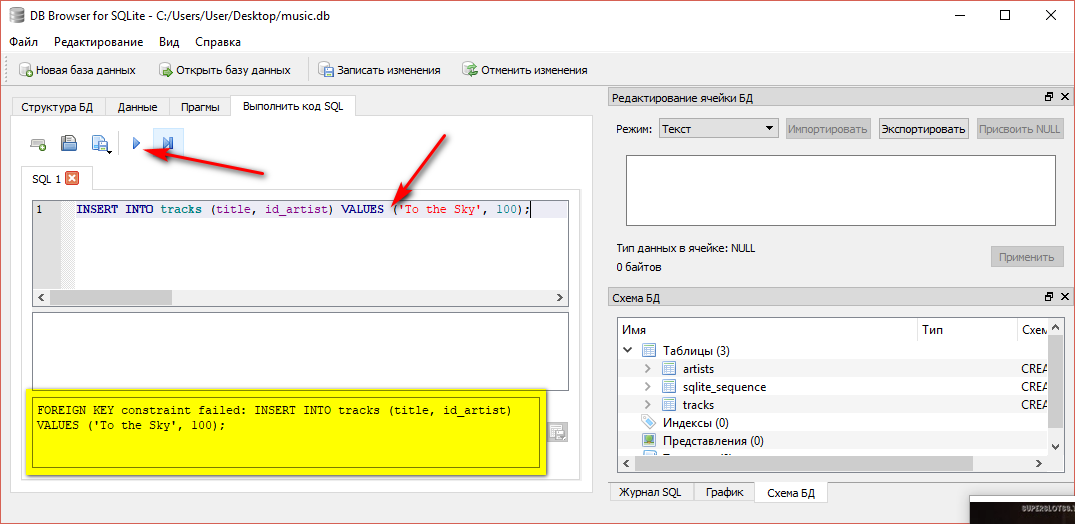
Запрос прошел успешно.

[](http://blog.harrix.org/wp-content/uploads/2017/06/2017-06-10_180108.png)

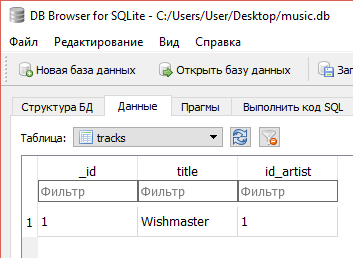
А теперь попробуем добавить запись с несуществующим номером исполнителя.

|  |  |
| --- | --- |
| 1 | INSERT INTO tracks (title, id\_artist) VALUES ('To the Sky', 100); |

А теперь запрос не пройдет из-за внешнего ключа.

[](http://blog.harrix.org/wp-content/uploads/2017/06/2017-06-10_180255.png)

И в таблице **tracks** мы увидим только одну запись.

[](http://blog.harrix.org/wp-content/uploads/2017/06/2017-06-10_180507.png)

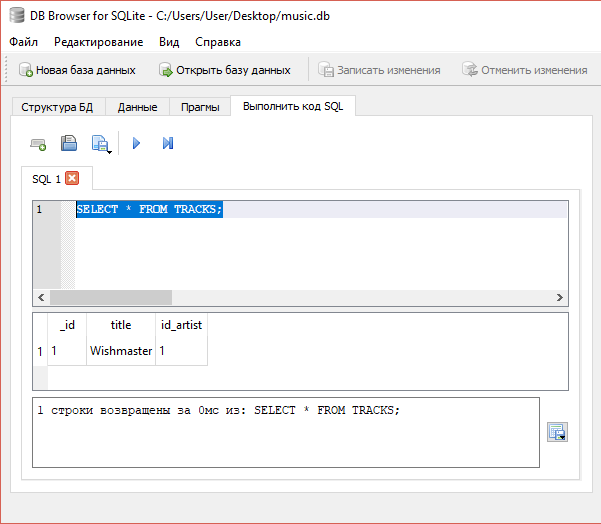
Обратите внимание на то, что в **SQLite** можно добавлять внешние ключи через **CREATE**. В «полноценных» БД можно добавлять внешние ключи через **ALTER**. Но в SQLite это сделать [нельзя](https://stackoverflow.com/questions/1884818/how-do-i-add-a-foreign-key-to-an-existing-sqlite-3-6-21-table).

|  |  |
| --- | --- |
| 1  2  3 | ALTER TABLE child ADD CONSTRAINT fk\_child\_parent                    FOREIGN KEY (parent\_id)                    REFERENCES parent(id); |

И напоследок покажу, как сделать выборку песен из БД так, чтобы вместо номеров песен мы видели названия самих исполнителей.

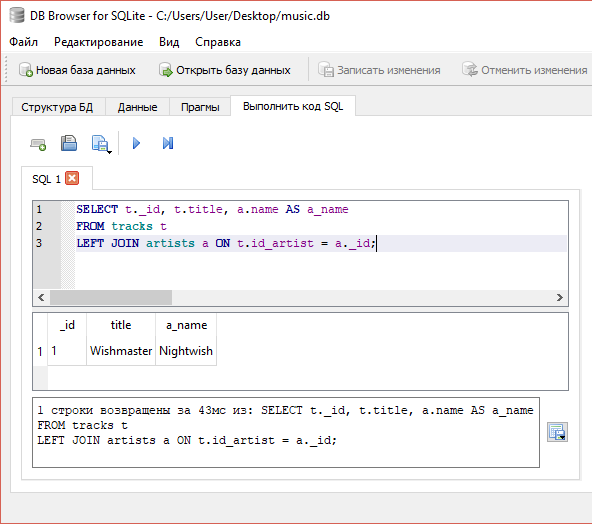
Этот пример стандартного вызова SELECT, когда мы увидим песни с номерами исполнителей.

|  |  |
| --- | --- |
| 1 | SELECT \* FROM TRACKS; |

[](http://blog.harrix.org/wp-content/uploads/2017/06/2017-06-10_181502.png)

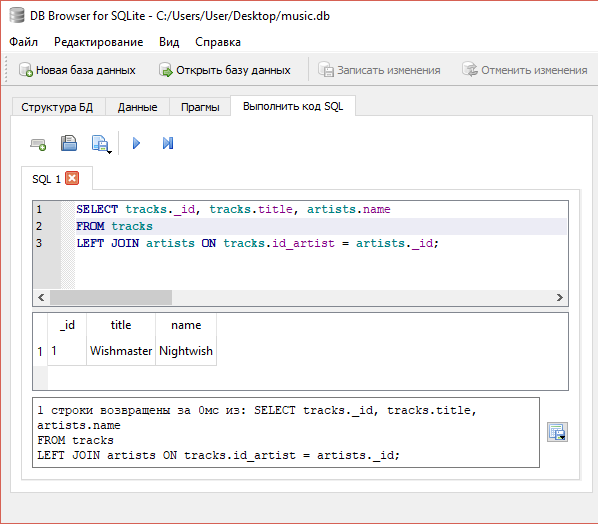
А вот пример кода, который вернет нам то, что нужно.

|  |  |
| --- | --- |
| 1  2  3 | SELECT t.\_id, t.title, a.name AS a\_name  FROM tracks t  LEFT JOIN artists a ON t.id\_artist = a.\_id; |

[](http://blog.harrix.org/wp-content/uploads/2017/06/2017-06-10_184220.png)

Более упрощенный код запроса.

|  |  |
| --- | --- |
| 1  2  3 | SELECT tracks.\_id, tracks.title, artists.name  FROM tracks  LEFT JOIN artists ON tracks.id\_artist = artists.\_id; |

[](http://blog.harrix.org/wp-content/uploads/2017/06/2017-06-10_184417.png)

Объяснение SQL объединений JOIN: LEFT/RIGHT/INNER/OUTER

Категория: / [DEV Блог](http://www.skillz.ru/dev/) / [PHP (LAMP)](http://www.skillz.ru/dev/php/)

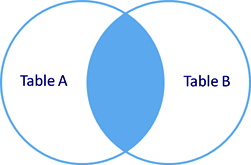
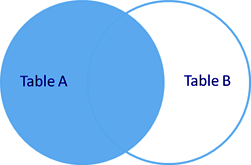
Разберем пример. Имеем две таблицы: пользователи и отделы.

U) users               D) departments  
id name       d\_id     id  name  
-- ----       ----     --  ----  
1  Владимир    1       1  Сейлз  
2  Антон       2       2  Поддержка  
3  Александр   6       3  Финансы  
4  Борис       2       4  Логистика  
5  Юрий        4

**SELECT** u.id, u.name, d.name **AS** d\_name  
**FROM** users u  
**INNER** **JOIN** departments d **ON** u.d\_id = d.id

Запрос вернет объединенные данные, которые пересекаются по условию, указанному в INNER JOIN ON <..>.   
В нашем случае условие <таблица\_пользователей>.<идентификатор\_отдела> должен совпадать с <таблица\_отделов>.<идентификатор>  
  
В результате *отсутствуют*:  
  
- пользователь Александр (отдел 6 - не существует)  
- отдел Финансы (нет пользователей) 

id name       d\_name  
-- --------   ---------  
1  Владимир    Сейлз  
2  Антон       Поддержка  
4  Борис       Поддержка  
3  Юрий        Логистика

  
  
*рис. Inner join*  
  
Внутреннее объединение INNER JOIN (синоним JOIN, ключевое слово INNER можно опустить).   
  
Выбираются только совпадающие данные из объединяемых таблиц.   
  
  
Чтобы получить данные, которые подходят по условию частично, необходимо использовать   
  
внешнее объединение - OUTER JOIN.   
  
Такое объединение вернет данные из обеих таблиц (совпадающие по условию объединения) ПЛЮС дополнит выборку оставшимися данными из внешней таблицы, которые по условию не подходят, заполнив недостающие данные значением NULL.  
  
  
  
*рис. Left join*  
  
Существует два типа внешнего объединения OUTER JOIN - LEFT OUTER JOIN и RIGHT OUTER JOIN.   
  
Работают они одинаково, разница заключается в том что LEFT - указывает что "внешней" таблицей будет находящаяся слева (в нашем примере это таблица users).   
Ключевое слово OUTER можно опустить. Запись LEFT JOIN идентична LEFT OUTER JOIN.

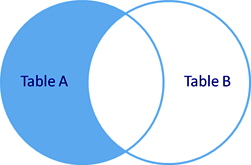
**SELECT** u.id, u.name, d.name **AS** d\_name  
**FROM** users u  
LEFT **OUTER** **JOIN** departments d **ON** u.d\_id = d.id

Получаем полный список пользователей и сопоставленные департаменты.

id      name          d\_name  
--      --------      ---------  
1       Владимир      Сейлз  
2       Антон         Поддержка  
3       Александр     NULL  
4       Борис         Поддержка  
5       Юрий          Логистика

Добавив условие 

**WHERE** d.id **IS** **NULL**

в выборке останется только 3#Александр, так как у него не назначен департамент.  
  
  
  
*рис. Left outer join с фильтрацией по полю*  
  
  
RIGHT OUTER JOIN вернет полный список департаментов (правая таблица) и сопоставленных пользователей.

**SELECT** u.id, u.name, d.name **AS** d\_name  
**FROM** users u  
RIGHT **OUTER** **JOIN** departments d **ON** u.d\_id = d.id

id      name        d\_name  
--      --------    ---------  
1       Владимир    Сейлз  
2       Антон       Поддержка  
4       Борис       Поддержка  
NULL    NULL        Финансы  
5       Юрий        Логистика

Дополнительно можно отфильтровать данные, проверяя их на NULL.

**SELECT** d.id, d.name  
**FROM** users u  
RIGHT **OUTER** **JOIN** departments d **ON** u.d\_id = d.id   
**WHERE** u.id **IS** **null**

В нашем примере указав WHERE u.id IS null, мы выберем департаменты, в которых не числятся пользователи. (3#Финансы)  
  
  
**Все примеры вы можете протестировать здесь:**  
  
[SQLFiddle](http://www.skillz.ru/r.php?url=http://sqlfiddle.com/#!2/659f69/1)  
  
  
**Cross/Full Join**  
  
FULL JOIN возвращает `объединение` объединений LEFT и RIGHT таблиц, комбинируя результат двух запросов.  
  
CROSS JOIN возвращает перекрестное (декартово) объединение двух таблиц. Результатом будет выборка всех записей первой таблицы объединенная с каждой строкой второй таблицы. Важным моментом является то, что для кросса не нужно указывать условие объединения.  
  
**Дублирование строк при использовании JOIN**  
  
При использовании объединения новички часто забывают что результирующая выборка может содержать дублирующиеся данные!  
Если вам нужна одна запись, делайте объединение с подзапросом

**SELECT** t1.\*, t2.\* **from** left\_table t1 left **join** (**select** \* **from** right\_table **where** some\_column = 1 **limit** 1) t2 **ON**t1.id = t2.join\_id

**Self Join**  
  
Выборка из одной и той же таблицы для нескольких условий.  
  
Рассмотрим задачку от яндекса:   
  
Есть таблица товаров.

**CREATE** **TABLE** `ya**\_**goods` (  
  `id` **int**(11) **unsigned** **NOT** **NULL** **AUTO\_INCREMENT**,  
  `name` **varchar**(64) **NOT** **NULL**,  
  **PRIMARY KEY** (`id`)  
) **ENGINE**=**InnoDB** **DEFAULT** **CHARSET**=utf8;  
**insert** **into** ya\_goods **values** (1, 'яблоки'), (2, 'яблоки') ,(3, 'груши'), (4,'яблоки'), (5, 'апельсины'), (6,'груши');

Она содержит следующие значения.

`id`    `name`  
1       Яблоки  
2       Яблоки  
3       Груши  
4       Яблоки  
5       Апельсины  
6       Груши

Напишите запрос, выбирающий уникальные пары `id` товаров с одинаковыми `name`, например:  
  
(1,2), (4,1), (2,4), (6,3)...  
  
При решении задачи необходимо учесть, что пары (x,y) и (y,x) — одинаковы.  
  
Решение:

**SELECT** g1.id id1, g2.id id2  
*-- CONCAT('(', LEAST(g1.id, g2.id), ',', GREATEST(g1.id, g2.id), ')') row*  
**FROM** ya\_goods g1   
**INNER** **JOIN** ya\_goods g2 **ON** g1.name = g2.name   
**WHERE** g1.id <> g2.id  
**GROUP BY** LEAST(g1.id, g2.id), GREATEST(g1.id, g2.id)  
**ORDER BY** g1.id;  
   
*-- или без группировки (быстрее)*  
   
**SELECT** **DISTINCT** CONCAT('(', LEAST(g1.id, g2.id), ',', GREATEST(g1.id, g2.id), ')') row  
**FROM** ya\_goods g1   
**INNER** **JOIN** ya\_goods g2 **ON** g1.name = g2.name   
**WHERE** g1.id <> g2.id

Объединяем таблицы ya\_goods по одинаковому полю `name`, группируем по уникальным idентификаторам и получаем результат.  
  
(1,2)(1,4)(2,4)(3,6)  
  
  
**Множественное объединение multi join**  
  
Пригодится нам, если необходимо выбрать более одного значения из таблиц для нескольких условий.  
  
Пример: набор вариантов (вес, объем) товаров.   
Продукты в таблице products, Варианты - таблица product\_options, Значения вариантов - таблица product2options  
Необходимо: фильтровать продукты по дате, и имеющимся вариантам

**CREATE** **TABLE**  `products` (  
  `id` **int**(11),  
  `title` **varchar**(255),  
  `created**\_**at` **datetime**  
)  
   
**CREATE** **TABLE** `product**\_**options` (  
  `id` **int**(11),  
  `name` **varchar**(255)  
)  
   
**CREATE** **TABLE** `product2options` (  
  `product**\_**id` **int**(11),  
  `option**\_**id` **int**(11),  
  `value` **int**(11)  
)

Тестовые данные

**INSERT** **INTO** `products` (`id`, `title`, `created**\_**at`) **VALUES**  
        (1, 'Кружка', '2009-01-17 20:00:00'),  
        (2, 'Ложка', '2009-01-18 20:00:00'),  
        (3, 'Тарелка', '2009-01-19 20:00:00');  
   
**INSERT** **INTO** `product**\_**options` (`id`, `name`) **VALUES**  
        (11, 'Вес'),  
        (12, 'Объем');  
   
**INSERT** **INTO** `product2options` (`product**\_**id`, `option**\_**id`, `value`) **VALUES**  
        (1, 11, 200),  
        (1, 12, 250),  
        (2, 11, 35),  
        (2, 12, 15),  
        (3, 11, 310),  
        (3, 12, 300),  
        (2, 11, 45),  
        (2, 12, 25);

Пример: выбрать товары,   
добавленные после 17/01/2009 в следующих вариантах:  
- вес=310, объем=300  
- вес=35, объем=15  
- вес=45, объем=25  
- вес=200, объем=250  
  
Просто перечислить условия вариантов в подзапросе/джоине через OR/AND не сработает,   
необходимо осуществить объединение таблиц вариантов равное количеству этих самых вариантов (у нас - 2: объем и вес)

**SELECT** p.\*, po1.name 'P1', p2o1.**value**, po2.name 'P2', p2o2.**value**       
   
**FROM** products p         
   
**INNER** **JOIN** product2options p2o1 **ON** p.id = p2o1.product\_id  
**INNER** **JOIN** product\_options po1  **ON** po1.id = p2o1.option\_id  
   
**INNER** **JOIN** product2options p2o2 **ON** p.id = p2o2.product\_id  
**INNER** **JOIN** product\_options po2  **ON** po2.id = p2o2.option\_id  
   
**WHERE** p.created\_at > '2009-01-17 21:00'   
  **AND** ( *-- тарелка#3*  
  p2o1.option\_id = 11 **AND** p2o1.**value** = 310   
  **AND** p2o2.option\_id = 12 **AND** p2o2.**value** = 300  
  **OR**  *-- ложка#2*  
  p2o1.option\_id = 11 **AND** p2o1.**value** = 35   
  **AND** p2o2.option\_id = 12 **AND** p2o2.**value** = 15             
  **OR**  *-- ложка#2*  
  p2o1.option\_id = 11 **AND** p2o1.**value** = 45   
  **AND** p2o2.option\_id = 12 **AND** p2o2.**value** = 25     
  **OR**  *-- кружка#1 не попадает по дате*  
  p2o1.option\_id = 12 **AND** p2o1.**value** = 250   
  **AND** p2o2.option\_id = 11 **AND** p2o2.**value** = 200  
  )    
;

Результ выборки:

id      title        created\_at                P1        value    P2            value  
2       Ложка        2009-01-18 20:00:00       Вес        35      Объем         15  
3       Тарелка      2009-01-19 20:00:00       Вес        310     Объем         300  
2       Ложка        2009-01-18 20:00:00       Вес        45      Объем         25  
   
-- не попадает по дате  
1       Кружка          2009-01-17 20:00:00     Объем   250     Вес             200

[Этот пример на SQLFiddle](http://www.skillz.ru/r.php?url=http://sqlfiddle.com/#!9/49866/2)  
  
**UPDATE и JOIN**  
  
Объединение можно использовать совместно с UPDATE.   
Например, имеем таблицу houses (id, title, area). Нужно выбрать title, если в нем встречается `число м2`, заменить поле area, если оно меньше. Т.к. в mysql отстутсутствует поддержка регулярных выражений, нужно немного поколдовать с locate и substr.  
В подзапросе выбираем интересующие нас данные, и в финальной стадии осуществляем обновление данных подходящий по критерию (p5 > area).

**UPDATE** houses base  
**INNER** **JOIN** (  
*-- Антарис аренда офиса 1594 м2, по ставке 12700 руб. м2/год -> 1594*  
**SELECT**  
        id,  
        @baseString := title title,  
        @areaTitleEnd := LOCATE(' м2', @baseString) **as** p2,  
        @tmpString := LTRIM(REVERSE(SUBSTR(@baseString, 1, @areaTitleEnd))) **as** p3,  
        @areaTitleBegin := LEFT(@tmpString, -1 + LOCATE(' ', @tmpString)) **as** p4,  
        @**value** := CAST(REVERSE(@areaTitleBegin) **as** **UNSIGNED**) **as** p5  
   
        **FROM** ga\_pageviews  
        **WHERE** title **like** '**%**м2**%**'  
) calc **USING** (`id`)  
**SET** base.area = calc.p5  
**WHERE** base.area < calc.p5

**DELETE и JOIN**  
  
Рассмотрим пример с удалением дубликатов. Есть таблица tableWithDups (id, email). Нужно удалить строки с одинаковыми email:

**DELETE** tableWithDups  
**FROM** tableWithDups  
**INNER** **JOIN** (  
        **SELECT** MAX(id) **AS** lastId, email  
        **FROM** tableWithDups  
        **GROUP BY** email  
        **HAVING** COUNT(\*) > 1  
        ) dups **ON** dups.email = tableWithDups.email  
**WHERE** tableWithDups.id < dups.lastId;

Последние два примера не совместимы с ANSI SQL, но работают в mySQL.  
  
За бортом статьи остались смежные объединениям (а также специфичные для определенных базданных темы):  
SELF JOIN, FULL OUTER JOIN, CROSS JOIN (CROSS [OUTER] APPLY), операции над множествами UNION [ALL], INTERSECT, EXCEPT и т.д.