Часть 11.4: Внешние ключи в базах данных SQLite: FOREIGN KEY в SQLite3

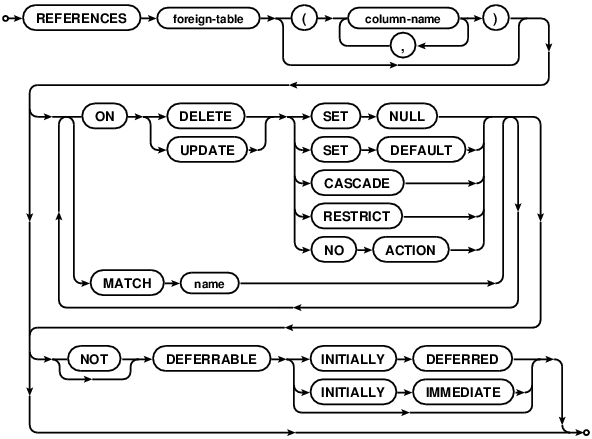
Работа внешних ключей в базах данных SQLite: FOREIGEN KEY и REFERENCE в SQLite3

СОДЕРЖАНИЕ СТАТЬИ:

* [1 Работа внешних ключей в базах данных SQLite: FOREIGEN KEY и REFERENCE в SQLite3](http://zametkinapolyah.ru/zametki-o-mysql/vneshnie-klyuchi-v-bazax-dannyx-sqlite.html#___SQLite_FOREIGEN_KEY_REFERENCE_SQLite3)
* [2 Реализация связи один ко многим в базах данных SQLite. Пример связи один ко многим и FOREIGEN KEY в SQLite](http://zametkinapolyah.ru/zametki-o-mysql/vneshnie-klyuchi-v-bazax-dannyx-sqlite.html#____SQLite____FOREIGEN_KEY_SQLite)
* [3 Реализация связи многие ко многим в базах данных SQLite3](http://zametkinapolyah.ru/zametki-o-mysql/vneshnie-klyuchi-v-bazax-dannyx-sqlite.html#____SQLite3)
* [4 Немного о логике внешних ключей в SQLite](http://zametkinapolyah.ru/zametki-o-mysql/vneshnie-klyuchi-v-bazax-dannyx-sqlite.html#___SQLite)

**Внешний ключ или FOREIGN KEY** – это [ограничение уровня таблицы в реляционных базах данных](http://zametkinapolyah.ru/zametki-o-mysql/ogranicheniya-urovnya-tablicy-v-bazax-dannyx-sqlite.html), в том числе и в базе данных SQLite3. Внешние ключи определяют правила, по которым будут связаны таблицы в базах данных SQLite. Но, кроме того, что внешние ключи определяют то, как будут связаны таблицы в базах данных SQLite3, они еще нужны для [обеспечения целостности данных в базах данных](http://zametkinapolyah.ru/zametki-o-mysql/obespechenie-celostnosti-dannyx-v-bazax-dannyx-sqlite3.html).

**В SQLite внешний ключ объявляется при помощи конструкции FOREIGN KEY**, а таблица, на которую ссылается внешний ключ указывается после ключевого слово REFERENCE. Обратите внимание: указывается не только таблица, но и столбец, на который идет ссылка. Ниже вы можете увидеть изображение, на котором показан синтаксис использования внешнего ключа в базах данных SQLite, вернее синтаксис конструкции REFERENCE.



*Синтаксис ограничения внешнего ключа в базах данных SQLite3*

Правила использования внешнего ключа не очень сложные, но давайте разберемся с тем, как реализован**внешний ключ в SQLite3** и его конструкции: **FOREIGEN KEY и REFEERENCE**. Обратите внимание: когда вы связываете таблицы при помощи внешнего ключа одна таблица является родительской, а вторая таблица является дочерней. Внешний ключ всегда ссылается на родительскую таблиц, другими словами конструкция FOREIGN KEY и REFERENCE указывается в дочерней таблице.

Подведем итог: внешний ключ в базах данных SQLite необходим для [реализации связей между таблицами](http://zametkinapolyah.ru/zametki-o-mysql/vidy-svyazej-mezhdu-tablicami-v-baze-dannyx-svyazi-v-relyacionnyx-bazax-dannyx-otnosheniya-kortezhi-atributy.html). FOREIGEN KEY позволяет реализовывать связи между таблицами в базах данных. Конструкция REFERENCE используется для указания ссылки на родительскую таблицу. Внешний ключ обеспечивает целостность данных между двумя таблицами и необходим для [нормализации базы данных](http://zametkinapolyah.ru/zametki-o-mysql/normalnye-formy-izbytochnost-dannyx-v-baze-dannyx-tranzitivnaya-zavisimost-proektirovanie-baz-dannyx.html). [Вторая нормальная форма](http://zametkinapolyah.ru/zametki-o-mysql/vtoraya-normalnaya-forma-2nf.html) и [третья нормальная форма](http://zametkinapolyah.ru/zametki-o-mysql/tretya-normalnaya-forma-3nf.html) не могут быть реализованы без внешнего ключа. Вернее будет сказать: мы можем организовать связи между таблицами без внешнего ключа, но проверка правил при этом выполняться не будет.

Давайте рассмотрим несколько практических **примеров реализации внешнего ключа в базах данных SQLite**. Мы реализуем два типа связи: связь один ко многим и связь многие ко многим.

Реализация связи один ко многим в базах данных SQLite. Пример связи один ко многим и FOREIGEN KEY в SQLite

Давайте **реализуем связь один ко многим при помощи внешнего ключа**, для этого воспользуемся конструкциями **FORIGEN KEY и REFERENCE**. Мы будем связывать при помощи внешнего ключа две таблицы: таблицу авторов и таблицу книг, поэтому давайте договоримся о допущение, что один автор может написать много книг, но у книги может быть только один автор.

Чтобы реализовать связь один ко многим, нам нужно воспользоваться конструкцией FOREIGEN KEY и REFERENCE при[создании таблицы](http://zametkinapolyah.ru/zametki-o-mysql/sozdanie-tablic-v-bazax-dannyx-sqlite.html) при помощи [команды CREATE](http://zametkinapolyah.ru/zametki-o-mysql/create-v-sqlite3-ddl.html).



|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27 | PRAGMA foreign\_keys=on;    CREATE TABLE books(    Id INTEGER PRIMARY KEY,    title TEXT NOT NULL,    count\_page INTEGER NOT NULL CHECK (count\_page >0),    price REAL CHECK (price >0),    auth\_id INTEGER NOT NULL,    FOREIGN KEY (auth\_id) REFERENCES auth(id)    );    CREATE TABLE auth(    id INTEGER PRIMARY KEY,    name TEXT NOT NULL,    age INTEGER  CHECK (age >16)    ); |

Здесь нам нужно дать пояснение к том, как мы создали **внешний ключ для базы данных**. Во-первых, в SQLite3 по умолчанию отключена поддержка внешних ключей, команда PRAGMA позволяет включить внешние ключи в базах данных SQLite. Во-вторых, помимо внешнего ключа наши таблицы имеют [ограничения уровня столбца](http://zametkinapolyah.ru/zametki-o-mysql/ogranicheniya-urovnya-stolbca-v-bazax-dannyx-sqlite.html). Третье, столбец который ссылается и столбец, на который ссылается, должны иметь одинаковый [тип данных](http://zametkinapolyah.ru/zametki-o-mysql/tema-5-tipy-dannyx-v-sqlite3.html), это закон реляционных баз данных.

Если столбцы будут иметь разные типы данных внешний ключ не будет работать. Скорее всего, другие [СУБД](http://zametkinapolyah.ru/zametki-o-mysql/arxitektura-subd-arxitektura-baz-dannyx-logicheskaya-struktura-subd-opisanie-dannyx-v-baze-dannyx-bazy-dannyx-sxema-dannyx.html) вам даже не дадут возможность создать такой внешний ключ, но [SQLite имеет динамическую типизацию данных](http://zametkinapolyah.ru/zametki-o-mysql/chast-5-1-obshhaya-informaciya-o-tipax-dannyx-v-sqlite3.html), поэтому внешний ключ будет создан.

Далее: конструкция FOREIGEN KEY объявляет о том, что столбец auth\_id является ссылкой, а конструкция REFERENCE указывает, что столбец auth\_id является ссылкой на столбец id таблицы auth. Таким нехитрым образом мы реализовали связь один ко многим в базе данных SQLite при помощи внешнего ключа.

Давайте теперь[добавим строки в таблицу](http://zametkinapolyah.ru/zametki-o-mysql/dobavlenie-dannyx-i-strok-v-tablicy-baz-dannyx-sqlite.html) auth нашей базы данных, это можно сделать [командой INSERT](http://zametkinapolyah.ru/zametki-o-mysql/chast-7-2-komanda-insert-v-sqlite3-operator-insert-v-sqlite3.html):



|  |  |
| --- | --- |
| 1  2  3  4  5  6  7 | INSERT INTO auth (id, name, age)    VALUES (1, ‘Джек Лондон’, 40);    INSERT INTO auth (id, name, age)    VALUES (2, ‘Лев Толстой’, 82); |

Добавлять данные при создании внешнего ключа и использовании FOREIGN KEY в нашем случае удобнее сперва в ту таблицу, на которую идет ссылка. Теперь добавим строки в таблицу books и укажем значения для нашего внешнего ключа.



|  |  |
| --- | --- |
| 1  2  3  4  5  6  7 | INSERT INTO books (id, title, count\_page, price, auth\_id)    VALUES (1, ‘Белый клык’, 287, 300.00, 1);    INSERT INTO books (id, title, count\_page, price, auth\_id)    VALUES (2, ‘Война и мир’, 806, 780.00, 2); |

Поскольку связь один ко многим, по нашей договоренности: один автор может написать много книг, добавим еще одно произведение Джека Лондона:



|  |  |
| --- | --- |
| 1  2  3 | INSERT INTO books (id, title, count\_page, price, auth\_id)    VALUES (3, ‘Зов предков’, 121, 160.00, 1); |

**А теперь давайте посмотрим, что нам дает связь один ко многим, реализованная благодаря внешнему ключу, и как действует ограничение FOREIGEN KEY**.



|  |  |
| --- | --- |
| 1  2  3 | INSERT INTO books (id, title, count\_page, price, auth\_id)    VALUES (4, ‘Белый клык’, 121, 160.00, 3); |

Этот [SQL запрос](http://zametkinapolyah.ru/zametki-o-mysql/sql-zaprosy-klyuchevye-slova-komandy-predlozheniya-i-sintaksis-yazyka-sql.html) INSERT не будет выполнен в SQLite3, поскольку действует ограничение внешнего ключа, мы получим ошибку: Error: FOREIGN KEY constraint failed. В таблице справочнике с авторами нет значения id = 3, SQLite это проверил и сказал нам, что мы не правы, что у  него есть FOREIGN KEY, который ничего не знает про автора с id = 3. Так обеспечивается целостность данных и так мы избавляемся от аномалии добавления данных в базах данных.

В нашем случае, если мы попробуем передать внешнему ключу значение NULL, то строка в таблицу не добавится, так как действует ограничение уровня столбца NOT NULL, но если бы этого ограничения не было, SQLite спокойно и честно добавил значение NULL, как ссылку в таблицу books, это может нарушить целостность ваших данных, поэтому будьте аккуратны, но это нельзя считать ошибкой, например, есть книги, автор которых неизвестен.

Попробуем изменить данные в таблицах, связанных при помощи внешнего ключа. Для этого воспользуемся [командой UPDATE](http://zametkinapolyah.ru/zametki-o-mysql/chast-7-3-komanda-update-v-sqlite3-operator-update-v-sqlite.html):



|  |  |
| --- | --- |
| 1 | UPDATE books SET auth\_id = 4 WHERE title = 'Белый клык'; |

SQLite не дала нам изменить значение auth\_id на 4, так как в таблице auth нет авторов с id = 4 и это**благодаря наличию внешнего ключа**. Ошибка в SQLite будет содержать следующий текст: Error: FOREIGN KEY constraint failed. В дальнейшем мы рассмотрим каскадное обновление данных при помощи команды ON UPDATE.

Нам осталось использовать последнюю команду из группы манипуляции данными, чтобы убедиться,**что внешние ключи в SQLite действительно обеспечивают целостность данных в базе данных**. Попробуем [удалить данные из таблицы](http://zametkinapolyah.ru/zametki-o-mysql/udalenie-dannyx-i-strok-iz-tablicy-bazy-dannyx-sqlite.html), для этого есть [команда DELETE](http://zametkinapolyah.ru/zametki-o-mysql/komanda-delete-v-sqlite3-operator-delete-dml.html).



|  |  |
| --- | --- |
| 1 | DELETE FROM auth WHERE name = 'Лев Толстой'; |

Сделать у нас это не получится, потому что сработает ограничение внешнего ключа, мы не можем удалить данные из таблицы справочника авторов, пока на них ссылаются значения из таблицы книг. Ошибка будет следующей: Error: FOREIGN KEY constraint failed.

Чтобы удалить данные из справочника, мы сперва должны удалить данные из таблицы которая ссылается на справочник, давайте это реализуем.



|  |  |
| --- | --- |
| 1  2  3 | DELETE FROM books WHERE auth\_id = 2;    DELETE FROM auth WHERE name = 'Лев Толстой'; |

Теперь мы удалили упоминание об авторе в таблице книг, а затем смогли удалить данные об авторе из таблицы auth, по-другому удалить данные из справочника мы не сможем: сперва удаляем данные из ссылающейся таблицы, а потом удаляем данные из таблицы, на которую ссылаемся – **это правило внешнего ключа**, которое обеспечивает целостность данных и защищает нас от аномалии удаления данных.

Про [команду SELECT](http://zametkinapolyah.ru/zametki-o-mysql/chast-7-1-komanda-select-v-sqlite3-operator-select-v-sqlite3.html) и то, как она работает с таблицами, которые используют внешние ключи, мы поговорим отдельно, так как это большая и требующая некоторых дополнительных усилий тема.

Сейчас нам важно понимать, что**конструкция FOREIGEN KEY REFERENCE** организует связь между таблицами [базы данных](http://zametkinapolyah.ru/zametki-o-mysql/bazy-dannyx-vidy-i-tipy-baz-dannyx-struktura-relyacionnyx-baz-dannyx-proektirovanie-baz-dannyx-setevye-i-ierarxicheskie-bazy-dannyx.html), а также то, что внешний ключ очень тесно связан с понятием [нормализации данных в базе данных](http://zametkinapolyah.ru/zametki-o-mysql/normalnye-formy.html). FOREIGEN KEY – это ограничения, которые заставляют SQLite проверять выполнение набора определенных правил, которые позволяют избежать аномалии модификации, удаления и обновления данных в таблицах базах данных.

Реализация связи многие ко многим в базах данных SQLite3

В принципе, мы уже сказали всё, что можно о **внешних ключах в базах данных SQLite и разобрали особенности работы FOREIGEN KEY и REFERENCE в SQLite3**. Раньше мы намеренно упрощали примеры, когда говорили про третью нормальную форму и когда разговаривали про особенности внешних ключей.

Давайте теперь исправим это упущение и будем считать, что одну книгу может написать много авторов и один автор может написать много книг. Реализуем связь многие ко многим в базах данных SQLite3 при помощи внешнего ключа.

Вы, наверное, помните, что для реализации связи многие ко многим нам необходимо создавать третью таблицу, ее можно назвать результирующей, а можно назвать и промежуточной. Мы назовем эту таблицу auth\_book.



|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35 | PRAGMA foreign\_keys=on;    CREATE TABLE books(    Id INTEGER PRIMARY KEY,    title TEXT NOT NULL,    count\_page INTEGER NOT NULL CHECK (count\_page >0),    price REAL CHECK (price >0)    );    CREATE TABLE auth(    id INTEGER PRIMARY KEY,    name TEXT NOT NULL,    age INTEGER  CHECK (age >16)    );    CREATE TABLE auth\_book (    auth\_id INTEGER NOT NULL,    books\_id INTEGER NOT NULL,    FOREIGN KEY (auth\_id) REFERENCES auth(id)    FOREIGN KEY (books\_id) REFERENCES books(id)    ); |

Никаких особых хитростей при реализации связи в базах данных SQLite, да и любых других реляционных базах данных нет. Мы создали третью таблицу, в которой есть два столбца и оба этих столбца являются внешними ключами. Столбец auth\_id является ссылкой на таблицу авторов, столбец books\_id – это ссылка на таблицу книг. Добавляем данные в таблицы командой INSERT.



|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39 | -- Сперва добавим несколько строк в таблицу книг    INSERT INTO books (id, title, count\_page, price)    VALUES (1, ‘Белый клык’, 287, 300.00);    INSERT INTO books (id, title, count\_page, price)    VALUES (2, ‘Война и мир’, 806, 780.00);    -- Затем добавим несколько авторов    INSERT INTO auth (id, name, age)    VALUES (1, ‘Джек Лондон’, 40);    INSERT INTO auth (id, name, age)    VALUES (2, ‘Лев Толстой’, 82);    -- Мы добавили книги и авторов из предыдущего примера    -- Давайте добавим новую книгу    INSERT INTO books (id, title, count\_page, price)    VALUES (3, ’12 стульев’, 516, 480.00);    -- И добавим авторов, так 12 стульев была написана в соавторстве    -- Затем добавим несколько авторов    INSERT INTO auth (id, name, age)    VALUES (3, ‘Илья Ильф’, 39);    INSERT INTO auth (id, name, age)    VALUES (4, ‘Евгений Петров’, 38); |

Мы подготовили таблицы, которые, на первый взгляд никак не связаны между собой, теперь давайте свяжем их связью многие ко многим, добавив нужные значения в столбцы FOREIGEN KEY таблицы auth\_book.



|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19 | -- Связываем таблицы auth и book    -- для этого наполним данными результирующую таблицу    INSERT INTO auth\_book (auth\_id, books\_id)    VALUES (1, 1);    INSERT INTO auth\_book (auth\_id, books\_id)    VALUES (2, 2);    INSERT INTO auth\_book (auth\_id, books\_id)    VALUES (3, 3);    INSERT INTO auth\_book (auth\_id, books\_id)    VALUES (4, 3); |

Хочу обратить ваше внимание на то, что SQLite и слышать не слышала про Льва Толстого, Джека Лондона и Ильфа с Петровым, она лишь проверяет работоспособность ссылок, ответственность за наполнение таблиц лежит на операторе. При таком подходе вы легко можете сказать, что Джек Лондон написал Войну и мир:



|  |  |
| --- | --- |
| 1  2  3  4  5 | -- Говорим SQLite о том, что Джек Лондон написал Войну и мир    INSERT INTO auth\_book (auth\_id, books\_id)    VALUES (1, 2); |

С точки логики SQLite никакой ошибки нет, с точки зрения человека – ошибка есть. **Внешние ключи никак не могут проверить смысл, который заложен в данных**, они лишь проверяют и ограничивают логику работы базы данных. Но не все так страшно, когда мы начнем работать с триггерами, мы посмотрим, как при помощи триггера сделать так, чтобы результирующая таблица при организации связи многие ко многим с помощью FOREIGN KEY заполнялась автоматически, это будет дополнительным уровнем обеспечения целостности данных в базах данных.

Немного о логике внешних ключей в SQLite

Когда мы рассматривали третью и вторую нормальные формы, мы ничего не упоминали про внешние ключи и про то, что они реализуют связи между таблицами. Тогда мы говорили, что справочник является дочерней таблицей, а вот таблица со ссылками (со столбцом FOREIGEN KEY) является родительской. С одной стороны, мы правильно говорим, до тех пор, пока не сталкиваемся со связью многие ко многим, ведь нельзя же результирующую таблицу называть родительской…

Поэтому в базах данных SQLite разработчики четко обозначили терминологию:

1. Родительская таблица — это таблица, на которую ссылается внешний ключ (FOREIGN KEY).
2. Дочерняя таблица — это таблица, к которой применяется ограничение внешнего ключа.
3. Родительский ключ — это один или несколько столбцов родительской таблицы, на которые ссылается внешний ключ (FOREIGN KEY).
4. Дочерний ключ — это один или несколько столбцов дочерней таблицы, значения которых ограничиваются внешним ключом и которые перечисляются после ключевой фразы REFERENCES.

На самом деле, вам нужно выбрать ту терминологию, которую вам легче воспринимать, мне легче воспринимать так: таблица справочник – дочерняя таблица, таблица, которая ссылается на справочник – родительская таблица.

Важно, чтобы другие вас понимали, благо, язык SQL универсальный и все понимают, что **FOREIGN KEY – это внешний ключ, который обеспечивает целостность данных**. FOREIGN KEY, как и[первичный ключ или PRIMARY KEY](http://zametkinapolyah.ru/zametki-o-mysql/chast-11-3-pervichnye-klyuchi-v-bazax-dannyx-sqlite-primary-key-ogranichenie-pervichnogo-klyucha.html), имеет синтаксис, который позволяет записывать данное ограничение, как ограничение уровня столбца.