Часть 11 3 Первичные ключи в базах данных SQLite: PRIMARY KEY. Ограничение первичного ключа

В теории баз данных ключ или ключевой атрибут — это столбец, который позволяет однозначно идентифицировать таблицу в базе данных. На практике, ни одна СУБД в мире не знает о таком столбце и не знает теории баз данных. На практике есть первичный ключ, который является правилом, которое SQLite ни когда не нарушит. SQLite будет следить всегда за тем, чтобы в столбце PRIMARY KEY были всегда уникальные и вечные значения.

В этой записи мы рассмотрим ограничения первичного ключа в базах данных SQLite в теории и на практике, так же узнаем, почему первичный ключ это не только правило базы данных, но еще и индекс, а так же поговорим о том, как реализовать составной первичный ключ для таблицы базы данных SQLite3

Первичный ключ, как ограничение уровня таблицы в базах данных SQLite. PRIMARY KEY в SQLIte3

СОДЕРЖАНИЕ СТАТЬИ:

* [1 Первичный ключ, как ограничение уровня таблицы в базах данных SQLite. PRIMARY KEY в SQLIte3](http://zametkinapolyah.ru/zametki-o-mysql/chast-11-3-pervichnye-klyuchi-v-bazax-dannyx-sqlite-primary-key-ogranichenie-pervichnogo-klyucha.html#_____SQLite_PRIMARY_KEY_SQLIte3)
* [2 Пример использования первичного ключа в SQLite. Пример PRIMARY KEY в SQLite](http://zametkinapolyah.ru/zametki-o-mysql/chast-11-3-pervichnye-klyuchi-v-bazax-dannyx-sqlite-primary-key-ogranichenie-pervichnogo-klyucha.html#___SQLite_PRIMARY_KEY_SQLite)
* [3 PRIMARY KEY, как индекс в базах данных SQLite3](http://zametkinapolyah.ru/zametki-o-mysql/chast-11-3-pervichnye-klyuchi-v-bazax-dannyx-sqlite-primary-key-ogranichenie-pervichnogo-klyucha.html#PRIMARY_KEY___SQLite3)
* [4 Составной первичный ключ в базах данных SQLite](http://zametkinapolyah.ru/zametki-o-mysql/chast-11-3-pervichnye-klyuchi-v-bazax-dannyx-sqlite-primary-key-ogranichenie-pervichnogo-klyucha.html#___SQLite)

Первичный ключ, это не только [ключевой атрибут](http://zametkinapolyah.ru/zametki-o-mysql/chast-3-3-klyuchi-i-klyuchevye-atributy-v-bazax-dannyx.html), который идентифицирует таблицу в базе данных и даже не только обязательное условие [второй нормальной формы](http://zametkinapolyah.ru/zametki-o-mysql/vtoraya-normalnaya-forma-2nf.html), а, соответственно, и [третьей нормальной формы](http://zametkinapolyah.ru/zametki-o-mysql/tretya-normalnaya-forma-3nf.html). **Первичный ключ или столбец PRIMARY KEY** – это [ограничение уровня таблицы](http://zametkinapolyah.ru/zametki-o-mysql/ogranicheniya-urovnya-tablicy-v-bazax-dannyx-sqlite.html) для любой [реляционной 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).

Ограничение уровня таблицы – это правило, которое никогда не должна нарушать ни одна СУБД и не должна никому давать это делать.**Первичный ключ или PRIMARY KEY в базах данных SQLite** – это столбец, значения которого должны быть уникальными и вечными, и что бы не произошло, что бы не случилось, SQLite никогда не нарушит правило уникальности и вечности первичного ключа.

Если мы задали первичный ключ для таблицы при помощи PRIMARY KEY, то, во-первых, мы сами должны заботиться об уникальности и вечности, во-вторых, SQLite будет делать это за нас. Если вы видите, что значения в столбце повторяющиеся и SQLite не дает вам их добавить, то, вероятнее всего, вы ошиблись при [проектировании базы данных](http://zametkinapolyah.ru/zametki-o-mysql/proektirovanie-baz-dannyx-teoriya-kompromissy-logika-zdravyj-smysl-i-predmetnaya-oblast.html) и назначили не тот столбец в качестве PRIMARY KEY.

Когда мы создаем первичный ключ, как ограничение таблицы, то вместо PRIMARY KEY мы вправе использовать конструкцию UNIQUE KEY, SQLite это позволяет и никаких проблем с таким подходом у вас не будет.

Помимо всего прочего, **первичный ключ в базе данных SQLite**, вернее столбец, который мы назначили, как PRIMARY KEY, является еще и индексом таблицы. Благодаря индексам таблицы операция выборки данных из базы данных при помощи [команды SELECT](http://zametkinapolyah.ru/zametki-o-mysql/chast-7-1-komanda-select-v-sqlite3-operator-select-v-sqlite3.html) происходит значительно быстрее, но место на диске [база 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) начинает занимать больше

**Вывод: первичный ключ в базах данных SQLite3 или PRIMARY KEY** – это правила, которые нельзя нарушать: правила уникальности и вечности значений столбца PRIMARY KEY. Так же первичный ключ является индексом таблицы в базе данных SQLite. Столбец, который объявлен, как PRIMARY KEY – это индекс, которому мы можем даже дать имя.

Пример использования первичного ключа в SQLite. Пример PRIMARY KEY в SQLite

Мы разобрались в теории, что собой представляет первичный ключ в реляционных базах данных, а также выяснили, что для создания первичного ключа необходимо использовать конструкцию PRIMARY KEY, тогда SQLite3 поймет, что данный столбец является первичным ключом, а его значения должны быть уникальными и вечными.

Давайте теперь на практике посмотрим, как первичный ключ может [обеспечить целостность данных в базе данных](http://zametkinapolyah.ru/zametki-o-mysql/obespechenie-celostnosti-dannyx-v-bazax-dannyx-sqlite3.html) и посмотрим, как в SQLite можно объявить столбец с ограничением PRIMARY KEY.

[Создадим таблицу в базе данных](http://zametkinapolyah.ru/zametki-o-mysql/sozdanie-tablic-v-bazax-dannyx-sqlite.html) с именем table1 при помощи [команды CREATE](http://zametkinapolyah.ru/zametki-o-mysql/create-v-sqlite3-ddl.html):

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9 | CREATE TABLE table1 (    a INTEGER PRIMARY KEY,    b TEXT NOT NULL,    c REAL    ); |

Первый столбец мы объявили **первичным ключом таблицы table1 при помощи ключевой фразы PRIMARY KEY**. Теперь давайте смотреть, как SQLite будет относиться к значениям, которые мы будем добавлять в первый столбец. Чтобы [добавить строку в таблицу](http://zametkinapolyah.ru/zametki-o-mysql/dobavlenie-dannyx-i-strok-v-tablicy-baz-dannyx-sqlite.html), нужно воспользоваться[командой INSERT](http://zametkinapolyah.ru/zametki-o-mysql/chast-7-2-komanda-insert-v-sqlite3-operator-insert-v-sqlite3.html).

|  |  |
| --- | --- |
| 1 | INSERT INTO table1 (NULL, ‘Hello, World!’, 0.2); |

Мы пробуем добавить значение NULL в столбец PRIMARY KEY, и у нас ничего не получается, первичный ключ не может иметь значение NULL, так как значение NULL нельзя ни с чем сравнить однозначно. А ошибку SQLite напишет такую: Error: near «NULL»: syntax error. Давайте теперь попробуем «обмануть» SQLite и добавим две строки:

|  |  |
| --- | --- |
| 1  2  3  4  5 | -- Добавляем две строки в таблицу, не указывая значение для столбца PRIMARY KEY    INSERT INTO table1 (b, c) VALUES (‘какой-то текст’, 0.5);    INSERT INTO table1 (b, c) VALUES (‘какой-то текст’, 0.7); |

SQLite добавляет эти строки в таблицу, хотя мы не указывали значение для первого столбца, давайте посмотрим на результат, сделав выборку данных из базы данных при помощи команды SELECT:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7 | -- Делаем выборку из базы данных и смотрим результат    SELECT\*FROM table1;    1|какой-то текст|0.5    2|какой-то текст|0.7 |

Как видите, в SQLite3 необязательно указывать [ограничение уровня столбца AUTOINCREMENT](http://zametkinapolyah.ru/zametki-o-mysql/ogranicheniya-urovnya-stolbca-v-bazax-dannyx-sqlite.html) для столбца с [аффинированным типом данных](http://zametkinapolyah.ru/zametki-o-mysql/chast-5-4-affinirovannye-tipy-dannyx-v-sqlite3.html) INTEGER, который объявлен, как **первичный ключ (PRIMARY KEY)**. А теперь попробуем добавить строку со значение a = 2 в таблицу table1 (это значение уже есть в таблице и SQLite не должна нам дать возможность добавить такую строку ).

|  |  |
| --- | --- |
| 1 | INSERT INTO table1 (2, ‘Hello, World!’, 0.2); |

SQLite заблокировала попытку добавления строки со значение a =2, так как правилом первичного ключа является его уникальность. Давайте подсунем SQLite в столбец «a» строку ‘2’. Тип данных будет другим, но, как мы знаем [SQLite – это СУБД с динамической типизацией данных](http://zametkinapolyah.ru/zametki-o-mysql/chast-5-1-obshhaya-informaciya-o-tipax-dannyx-v-sqlite3.html):

|  |  |
| --- | --- |
| 1 | INSERT INTO table1 (‘2’, ‘Hello, World!’, 0.2); |

Да, та же самая ошибка: Error: near «0.2»: syntax error. В данном случае, перед тем, как записать значение в таблицу SQLite его преобразовало в число, так как мы объявили для столбца «а» [класс данных](http://zametkinapolyah.ru/zametki-o-mysql/klassy-dannyx-vmesto-tipov-dannyx-v-sqlite3.html) INTEGER, а затем сравнила значения из таблицы с тем, что мы хотели добавить, после чего заблокировала операцию добавления данных.

Теперь давайте попробуем изменить значение столбца «а». Для изменения данных в базе данных SQLite есть специальная [команда UPDATE](http://zametkinapolyah.ru/zametki-o-mysql/chast-7-3-komanda-update-v-sqlite3-operator-update-v-sqlite.html), воспользуемся ей:

|  |  |
| --- | --- |
| 1  2  3 | UPDATE table1 SET a = 2 WHERE c = 0.5;    UPDATE table1 SET a = 3 WHERE c = 0.7; |

Когда мы первый раз используем команду UPDATE, SQLite заблокирует нам эту операцию, так как первичный ключ должен быть уникальным, а значение 2 уже есть в столбце «а». Второй UPDATE будет выполнен, так как наша таблица еще не связана с другими таблицами и здесь не произойдет нарушение правила уникальности. Убедимся в этом при помощи команды SELECT:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7 | -- Делаем выборку, чтобы убедиться, что второй UPDATE сработал    SELECT \* FROM table1;    1|какой-то текст|0.5    3|какой-то текст|0.7 |

Если интересно, то при выполнении первого UPDATE, SQLite выдала ошибку: Error: UNIQUE constraint failed: table1.a. Но давайте немного изменим пример, пусть первичный ключ нашей таблицы будет с аффинированным типом данных TEXT. Создадим таблицу table2:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9 | CREATE TABLE table2 (    a TEXT PRIMARY KEY,    b TEXT NOT NULL,    c REAL    ); |

Структура таблицы та же, что и в первом случае, но теперь **столбец PRIMARY KEY** имеет класс данных TEXT. Выполним первый INSERT из примера с типом данных INTEGER:

|  |  |
| --- | --- |
| 1 | INSERT INTO table2 (NULL, ‘Hello, World!’, 0.2); |

**Ограничение первичного ключа сработало**, мы получили ошибку, а это значит, что SQLite не добавила значение NULL в столбец PRIMARY KEY с типом данных TEXT.Выполним второй INSERT, не указав никаких значений для первичного ключа:

|  |  |
| --- | --- |
| 1 | INSERT INTO table2 (b, c) VALUES (‘какой-то текст’, 0.5); |

SQLite добавила строку в базу данных, несмотря на то, что мы не указывали никаких значений для столбца PRIMARY KEY, давайте сделаем SELECT:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7 | SELECT \* FROM table2;    |какой-то текст|0.5    SELECT typeof(a), typeof(b), typeof(c) FROM table2;    null|text|real |

Будьте аккуратны, если вы объявляете первичным ключом столбец с типом данных TEXT при использовании SQLite3, если вы забудете добавить значение, то автоматически добавится значение NULL, в этом случае нужно пользоваться [ограничение уровня столбца NOT NULL](http://zametkinapolyah.ru/zametki-o-mysql/ogranicheniya-urovnya-stolbca-v-bazax-dannyx-sqlite.html), чтобы избежать таких ошибок.

А теперь в таблицу table2, попробуем добавить две строки:

|  |  |
| --- | --- |
| 1  2  3 | INSERT INTO table2 (a,b,c) VALUES (5, ‘Hello, World!’, 0.2);    INSERT INTO table2 VALUES (‘5’, ‘Hello, World!’, 55); |

Первая строка будет добавлена, во второй строке мы получим ошибку. Это произошло из-за того, что SQLite перед тем, как добавить первую строку, преобразовал число 5 в строку ‘5’, поэтому на вторую строку было наложено правило уникальности первичного ключа.

А теперь давайте создадим таблицу table3 с первичным ключом PRIMARY KEY, для которого мы не станем указывать тип данных и попробуем повторить последний эксперимент:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17 | -- Создаем таблицу с первичным ключом, но без типов данных    CREATE TABLE table3 (    a PRIMARY KEY,    b NOT NULL,    c    );    -- Добавляем в эту таблицу две строки    INSERT INTO table3 VALUES (‘5’, ‘Hello, World!’, 55);    INSERT INTO table3 (a,b,c) VALUES (5, ‘Hello, World!’, 0.2); |

Все команды будут выполнены успешно, так как для столбца, объявленного, как PRIMARY KEY не был указан аффинированный тип данных, а значение 5 и ‘5’ разные значения, так как первое – это число, а второе – строка.

PRIMARY KEY, как индекс в базах данных SQLite3

Мы утверждаем, что **первичный ключ – это ограничение уровня таблицы**, но столбец PRIMARY KEY мы объявляем, как ограничение уровня столбца, на самом деле, такая форма записи не совсем правильная. Давайте разберемся, почему это так и как нужно делать правильно. Во-первых, стоит сказать, что ограничения уровня таблицы – это такие же объекты баз данных, как и сами таблицы и мы можем им давать имена, которые должны быть уникальны во всей базе данных.

А теперь давайте объявим столбец PRIMARY KEY, как ограничение уровня таблицы:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11 | CREATE TABLE table4(    a INTEGER,    b TEXT NOT NULL,    c REAL,    PRIMARY KEY (a)    ); |

Напишите команду .schema, чтобы увидеть, какой запрос создал данную таблицу. Но, мы утверждали, что первичный ключ – это ни что иное, как индекс таблицы в базе данных SQLite3 и что у столбца PRIMARY KEY может быть имя, давайте этом посмотрим:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11 | CREATE TABLE table5(    a INTEGER,    b TEXT NOT NULL,    c REAL,    CONSTRAINT ixpk PRIMARY KEY (a)    ); |

Теперь мы создали первичный ключ с именем ixpk, этот первичный ключ является индексом таблицы базы данных table5, в этом легко убедиться, воспользуйтесь командой SELECT, которая покажет все индексы в базе данных SQLite:

|  |  |
| --- | --- |
| 1 | SELECT \* FROM sqlite\_master WHERE type = 'index'; |

Мы создали первичный ключ, объявив столбец «а», как ограничение уровня таблицы при помощи конструкции CONSTRAINT ixpk PRIMARY KEY (a) после объявления всех столбцов, но, в то же время, мы создали индекс с именем ixpk для таблицы table5.

Составной первичный ключ в базах данных SQLite

Бывают такие ситуации, когда в таблице нет столбца, в котором значения были бы уникальные и при этом нежелательно создавать суррогатный ключ, который бы генерировала СУБД. В этом случае используются **составные первичные ключи**. Создать составной первичный ключ в базе данных SQLite очень просто, давайте посмотрим на пример:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11 | CREATE TABLE table6(    a INTEGR NOT NULL,    b TEXT,    c REAL,    CONSTRAINT new\_pk PRIMARY KEY (a, b, c)    ); |

Мы помним, что первичный ключ должен обладать двумя параметрами: уникальностью и вечностью. По логике вещей, если мы создаем составной первичный ключ, SQLite должна заботиться об уникальности всех трех значений в сумме, а не по отдельности, давайте это проверим:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7 | INSERT INTO table6 (a,b,c) VALUES (5, ‘Hello, World!’, 0.2);    INSERT INTO table6 (a,b,c) VALUES (5, ‘Hello, World!’, 0.8);    INSERT INTO table6 (a,b,c) VALUES (25, ‘Hello, World!’, 0.2);    INSERT INTO table6 (a,b,c) VALUES (5, ‘Hello, World!’, 0.2); |

Все команды INSERT, кроме последней, будут выполнены, последний [SQL запрос](http://zametkinapolyah.ru/zametki-o-mysql/sql-zaprosy-klyuchevye-slova-komandy-predlozheniya-i-sintaksis-yazyka-sql.html) INSERT выполнен не будет, так как нарушается правило уникальности, мы же создавали составной первичный ключ, поэтому SQLite проверяет на уникальность все три значения и, если все три значения вместе являются не уникальными, то SQLite не дает добавить такую строку в таблицу.