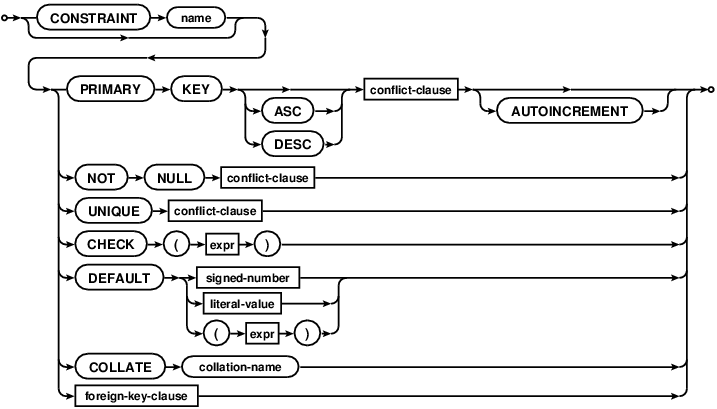
Часть 11 1 Ограничения уровня столбца в базах данных SQLite

Виды ограничений уровня столбца в базах данных SQLite

СОДЕРЖАНИЕ СТАТЬИ:

* [1 Виды ограничений уровня столбца в базах данных SQLite](http://zametkinapolyah.ru/zametki-o-mysql/ogranicheniya-urovnya-stolbca-v-bazax-dannyx-sqlite.html#____SQLite)
* [2 Типы данных, как ограничение уровня столбца в базах данных SQLite3](http://zametkinapolyah.ru/zametki-o-mysql/ogranicheniya-urovnya-stolbca-v-bazax-dannyx-sqlite.html#_____SQLite3)
* [3 Значение по умолчанию, как ограничение уровня столбца в базах данных SQLite. Ограничение DEFAULT в SQLite](http://zametkinapolyah.ru/zametki-o-mysql/ogranicheniya-urovnya-stolbca-v-bazax-dannyx-sqlite.html#_____SQLite_DEFAULT_SQLite)
* [4 Значение NOT NULL, как ограничение уровня столбца в базах данных SQLite3. Ограничение NOT NULL в SQLite](http://zametkinapolyah.ru/zametki-o-mysql/ogranicheniya-urovnya-stolbca-v-bazax-dannyx-sqlite.html#_NOT_NULL____SQLite3_NOT_NULL_SQLite)
* [5 Значение UNIQUE, как ограничение уровня столбца в базах данных SQLite. Ограничение UNIQUE в SQLite](http://zametkinapolyah.ru/zametki-o-mysql/ogranicheniya-urovnya-stolbca-v-bazax-dannyx-sqlite.html#_UNIQUE____SQLite_UNIQUE_SQLite)
* [6 Значение AUTOINCREMENT, как ограничение уровня столбца в базах данных SQLite. Ограничение AUTOINCREMENTв SQLite3.](http://zametkinapolyah.ru/zametki-o-mysql/ogranicheniya-urovnya-stolbca-v-bazax-dannyx-sqlite.html#_AUTOINCREMENT____SQLite_AUTOINCREMENT_SQLite3)
* [7 Значение CHECK, как ограничение уровня столбца в базах данных SQLite. Ограничение CHECKв SQLite3.](http://zametkinapolyah.ru/zametki-o-mysql/ogranicheniya-urovnya-stolbca-v-bazax-dannyx-sqlite.html#_CHECK____SQLite_CHECK_SQLite3)

Мы уже коротко рассмотрели ограничения уровня столбца, когда разговаривали про [обеспечение целостности данных в базах данных SQLite](http://zametkinapolyah.ru/zametki-o-mysql/obespechenie-celostnosti-dannyx-v-bazax-dannyx-sqlite3.html), давайте теперь более подробно поговорим **о реализации ограничений уровня столбца в базах данных под управлением SQLite3**.



*Синтаксис ограничения уровня столбца в базах данных SQLite*

Сверху на рисунке изображен синтаксис создания ограничений уровня столбца при [создании таблицы в SQLite3](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). Хочу обратить ваше внимание, что мы бы не стали относить ограничение первичного ключа PRIMARY KEY к ограничениям уровня столбца. Так же стоит заметить, что конструкция CONSTRAINT name является необязательной и ее часто опускают при создании таблиц в базах данных SQLite.

Давайте перечислим ограничения уровня столбца и дадим короткую характеристику каждому ограничению:

1. [Тип данных](http://zametkinapolyah.ru/zametki-o-mysql/tema-5-tipy-dannyx-v-sqlite3.html) столбца является ограничением уровня столбца, хоть и не указан на рисунке сверху. На самом деле в [SQLite нет типов данных](http://zametkinapolyah.ru/zametki-o-mysql/chast-5-1-obshhaya-informaciya-o-tipax-dannyx-v-sqlite3.html), это понятие заменено здесь на [класс данных](http://zametkinapolyah.ru/zametki-o-mysql/klassy-dannyx-vmesto-tipov-dannyx-v-sqlite3.html), а столбцу мы задаем [аффинированный тип данных](http://zametkinapolyah.ru/zametki-o-mysql/chast-5-4-affinirovannye-tipy-dannyx-v-sqlite3.html), который используется[для сравнения](http://zametkinapolyah.ru/zametki-o-mysql/sravnenie-dannyx-v-sqlite3-poryadok-sortirovki-v-sqlite3.html). Тип данных в 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) ограничивает типы данных для значений, которые могут храниться в столбце.
2. **Ограничение NOT NULL является ограничений уровня столбца**. Если вы задали столбцу ограничение NOT NULL, то SQLite при [добавлении строк в таблицу базы данных](http://zametkinapolyah.ru/zametki-o-mysql/dobavlenie-dannyx-i-strok-v-tablicy-baz-dannyx-sqlite.html)командой [INSERT](http://zametkinapolyah.ru/zametki-o-mysql/chast-7-2-komanda-insert-v-sqlite3-operator-insert-v-sqlite3.html) не даст вам «забыть» записать в этот столбец значение отличное от NULL.
3. Ограничению UNIQUE заставляет SQLite проверять уникальность значений в столбце таблицы. Если вы задали для столбца ограничение UNIQUE, то SQLite не даст вам записать в него повторяющееся значение.
4. Ограничение CHECK, данное ограничение задает диапазон значений, которые могут храниться в столбце, например, в столбце зарплата вы можете установить ограничение, что зарплата не может быть меньше МРОТ при помощи ограничения CHECK.
5. **Ограничение уровня столбца DEFAULT**позволяет задать значение по умолчанию, которое будет добавляться автоматически, если вы забудете про столбец при добавлении новых строк в таблицу.
6. Ограничение уровня столбца COLLATE позволяет задать способ сравнения данных в SQLite,**на первый взгляд COLLATE не является ограничение уровня столбца**, но, когда мы[удаляем данные из таблицы](http://zametkinapolyah.ru/zametki-o-mysql/udalenie-dannyx-i-strok-iz-tablicy-bazy-dannyx-sqlite.html), то обычно используем клаузулу WHERE для фильтрации данных, таким образом мы выбираем, какие строки хотим удалить из таблицы [командой DELETE](http://zametkinapolyah.ru/zametki-o-mysql/komanda-delete-v-sqlite3-operator-delete-dml.html).

Ниже мы на примерах рассмотрим каждый из перечисленных видов ограничения уровня столбца.

Типы данных, как ограничение уровня столбца в базах данных SQLite3

Мы уже упоминали о том, что тип данных является ограничением уровня столбца в [реляционных базах 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 еще раньше, когда мы говорили о типах данных, мы говорили что SQLite– это СУБД с динамической типизацией данных. С одной стороны: это дает плюсы, например, при создании таблицы вы можете не указывать тип данных для столбца, но с другой стороны, SQLite динамически определяет с каким типом данных значение будет храниться в столбце, а это уже может [нарушить целостность данных](http://zametkinapolyah.ru/zametki-o-mysql/obespechenie-celostnosti-dannyx-v-bazax-dannyx-sqlite3.html) и привести к самым неожиданным последствиям и ошибкам.

Давайте на примере посмотрим, почему типы данных являются ограничение уровня столбца или не являются. Для этого мы создадим простую таблицу командой CREATE.

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11 | -- Создаем таблицу и объявляем тип данных для столбца    CREATE TABLE table1(    a INTEGER,    b REAL,    c TEXT    ); |

Мы создали таблицу с тремя столбцами и указали для каждого столбца тип данных, надеясь, что SQLite будет ограничивать ввод данных с неправильно указанным типом. Теперь давайте добавим несколько строк в таблицу при помощи команды INSERT.

|  |  |
| --- | --- |
| 1  2  3  4  5 | -- Добавление строк в таблицу    INSERT INTO table1    VALUES (20, 2.02345, ‘текст’); |

Мы, как честные разработчики, в каждый столбец добавили значение с тем типом данных, который был указан при создании таблицы. А теперь убедимся, что данные были записаны верно, сделав выборку данных из таблицы и проверив тип данных значения при помощи [команды SELECT](http://zametkinapolyah.ru/zametki-o-mysql/chast-7-1-komanda-select-v-sqlite3-operator-select-v-sqlite3.html) и функции typeof.

|  |  |
| --- | --- |
| 1  2  3 | -- Проверям типы данных для столбца    SELECT typeof(a), typeof(b), typeof(c) FROM table1; |

Выполнив команду SELECT, мы увидим типы данных значений в столбце:

|  |  |
| --- | --- |
| 1 | integer|real|text |

Теперь давайте удалим данные из таблицы при помощи команды DELETE, но добавлять новые данные в таблицу будем нечестно:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7 | -- Удаляем данные из таблицы и добавляем новые данные с ошибкой    DELETE FROM table1;    INSERT INTO table1;    VALUES (20.12321312, ‘3212.02’, 158); |

Теперь выполним две команды SELECT, первой мы посмотрим какие значения сохранились, второй мы посмотрим, какой тип данных у столбцов.

|  |  |
| --- | --- |
| 1  2  3  4  5 | -- Проверяем какие сохранились значения и какой тип данных у этих значений    SELECT \* FROM table1;    SELECT typeof(a), typeof(b), typeof(c) FROM table1; |

Результат будет таким:

|  |  |
| --- | --- |
| 1  2  3 | 20.12321312|3212.02|158    real|real|text |

Мы видим, что SQLite нам в принципе дала добавить значение с отличным типом данных от того, что мы указывали при создании таблицы, поэтому **тип данных является слабым ограничением уровня столбца**, но давайте усложним эксперимент:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7 | -- Удаляем данные из таблицы и добавляем новые данные с ошибкой    DELETE FROM table1;    INSERT INTO table1    VALUES (’12 негритят’, ‘текст’, ‘честно добавляем текст’); |

SQLite спокойно добавляет данные с другим типом данных в столбцы, поэтому в базах данных SQLite можно забыть о том, что тип данных является ограничением уровня столбца, в отличи от MySQL или PostgreSQL, эти СУБД нам не дадут записать значение с другим типом данных в столбец.

Убедимся, что всё было добавлено:

|  |  |
| --- | --- |
| 1  2  3  4  5 | -- Проверяем какие сохранились значения и какой тип данных у этих значений    SELECT \* FROM table1;    SELECT typeof(a), typeof(b), typeof(c) FROM table1; |

А вот и результат:

|  |  |
| --- | --- |
| 1  2  3 | 12 негритят|текст|честно добавляем текст    text|text|text |

**Вывод: в SQLite3 тип данных не является ограничением уровня столбца, типы данных в SQLite нужны лишь для сравнения данных**.

Значение по умолчанию, как ограничение уровня столбца в базах данных SQLite. Ограничение DEFAULT в SQLite

Мы выяснили, что в базах данных SQLite тип данных не является ограничение уровня столбца, давайте посмотрим, что дает нам **ограничение столбца DEFAULT** и почему значение по умолчанию является ограничением уровня столбца.

Создадим простую таблицу table2:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13 | -- Создаем таблицу с именем table2    -- и задаем столбцу значение по умолчанию    CREATE TABLE table2(    a INTEGER DEFAULT 15,    b TEXT,    c TEXT    ); |

Мы создали таблицу и задали значение по умолчанию для первого столбца при помощи ограничения DEFAULT.

Теперь давайте добавим данные в таблицу командой INSERT:

|  |  |
| --- | --- |
| 1  2  3  4  5 | -- добавление данных в таблицу    INSERT INTO table2    VALUES (12, ‘текст первый’, ‘текст второй’); |

Мы честно добавили значения во все три столбца таблицы table2, но если мы забудем добавить значение для первого столбца, SQLite подставит в него значение 15 из-за ограничения DEFAULT, которое мы добавили.

|  |  |
| --- | --- |
| 1  2  3 | INSERT INTO table2 (b, c)    VALUES (‘текстпервый’, ‘текствторой’); |

Проверку на то, что ограничение DEFAULT сработало, можно осуществить при помощи оператора SELECT:

|  |  |
| --- | --- |
| 1  2  3 | -- Проверяем работу ограничения DEFAULT    SELECT \* FROM table2; |

Результат выборки из-за ограничения DEFAULT будет таким:

|  |  |
| --- | --- |
| 1  2  3 | 12|текст первый|текст второй    15|текст первый|текст второй |

Заметьте, при втором INSERT мы не добавляли значение к первому столбцу, это за нас сделало**ограничение уровня столбца DEFAULT**.

**Вывод: ограничение уровня столбца DEFAULT работает в SQLite и помогает забывчивым пользователям, добавляя значения, если оно не указано.**

Значение NOT NULL, как ограничение уровня столбца в базах данных SQLite3. Ограничение NOT NULL в SQLite

Если ограничение DEFAULT позволяет задать значение по умолчанию для столбца, то ограничение NOT NULL не дает пользователю добавить значение NULL. Ограничение уровня столбца NOT NULL исключает намеренные и ненамеренные ошибки ввода, например, пользователь намеренно пытается внести в базу данных значение NULL, а может, оператор просто забыл вписать значение в поле. И в первом, и во втором случае SQLite выдаст ошибку и скажет, что для добавления данных в таблицу значение должно отличаться от NULL.

Давайте в этом убедимся, создайте таблицу table3:

|  |  |
| --- | --- |
| 1  2  3 | -- создаем таблицу с ограничение столбца NOT NULL    CREATE TABLE table3 (A NOT NULL, b, c); |

А теперь давайте добавим данные в таблицу:

|  |  |
| --- | --- |
| 1  2  3  4  5 | -- добавляем данные    INSERT INTO table3 (a, b)    VALUES (‘значение’, NULL); |

А теперь проверим, что в итоге получилось командой SELECT:

|  |  |
| --- | --- |
| 1  2  3 | -- Проверяем значения в таблице    SELECT typeof(a), typeof(b), typeof(c) FROM table3; |

Результат будет следующим:

|  |  |
| --- | --- |
| 1 | text|null|null |

А теперь посмотрим, как SQLite отработает **ограничение уровня столбца NOT NULL**, давайте попробуем намеренно добавить значение NULL в первый столбец, но перед этим не забудьте удалить данные из table3:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7 | -- Проверка ограничения NOTNULLв SQLite3    DELETE FROM table3;    INSERTINTOtable3 (a, b, c)    VALUES (NULL, 100, ‘какой-то текст’); |

В результате мы увидим, что **ограничение столбца NOT NULL сработало**, а SQLite3 выдала ошибку: Error: NOT NULL constraint failed: table3.A. Но что будет, если мы «забудем» добавить значение в первый столбец и как сработает ограничение NOT NULL в базе данных.

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7 | -- Мы «забыли» добавить значение    DELETEFROMtable3;    INSERTINTOtable3 (b, c)    VALUES (100, ‘какой-то текст’); |

SQLite сработала верно, выдав ошибку: Error: NOTNULLconstraintfailed: table3.A.Так как ограничение столбца NOT NULL сработало, а в том случае, когда мы пытаемся добавить данные, не указав значение для столбца по умолчанию записывается значение NULL.

**Вывод: ограничение уровня столбца NOTNULLв базах данных SQLiteзаботится о поддержании целостности данных и не дает намеренно или случайно добавлять значения NULLв таблицы.**

Значение UNIQUE, как ограничение уровня столбца в базах данных SQLite. Ограничение UNIQUE в SQLite

Ограничение UNIQUE в SQLite может быть ограничением уровня столбца, а может являться псевдонимом ограничения уровня таблицы, когда мы пытаемся создать [первичный ключ](http://zametkinapolyah.ru/zametki-o-mysql/chast-3-3-klyuchi-i-klyuchevye-atributy-v-bazax-dannyx.html) в базе данных. Сейчас мы поговорим про **UNIQUE, как о ограничении уровня столбца в базах данных SQLite**.

Ограничение UNIQUE не даст вам возможность добавить повторяющиеся значения в таблицу базы данных. Давайте посмотрим в действие ограничение уровня столбца UNIQUE. Создайте таблицу table4.

|  |  |
| --- | --- |
| 1  2  3  4  5 | -- Создаем таблицу с ораничением    -- уровня столбца UNIQUE    CREATE TABLE table4 (a UNIQUE, b, c); |

А теперь давайте наполним таблицу данными:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7 | -- Добавляем данные в таблицу    INSERT INTO table4 (a, b, c) VALUES (NULL, 100, ‘какой-тотекст’);    INSERT INTO table4 (a, b, c) VALUES (NULL, 58, ‘какой-тотекст 2’);    INSERT INTO table4 (a, b, c) VALUES (0.21312312, 199, ‘какой-тотекст 3’); |

SQLite честно добавит три строки в таблицу и ограничение UNIQUE не сработает на значение NULL, поскольку NULL – это всегда уникальное значение, которое не равно ничему, даже самому себе. А теперь попробуем добавить четвертую строку в [таблицу базы данных](http://zametkinapolyah.ru/zametki-o-mysql/rabota-s-tablicami-v-bazax-dannyx-sqlite3.html):

|  |  |
| --- | --- |
| 1 | INSERT INTO table4 (a, b, c) VALUES (0.21312312, 105, ‘какой-то текст 4’); |

SQLite3 не даст нам добавить эту строку в таблицу и выдаст ошибку: Error: UNIQUEconstraintfailed: table4.a. Поскольку сработает ограничение уровня столбца UNIQUE, так как добавляемое значение в столбец «а» не уникально.

**Вывод: ограничение уровня столбца UNIQUE обеспечивает поддержание целостности данных в том случае, когда в столбце должны храниться уникальные значения.**Ограничение UNIQUE не распространяется на значение NULL и это правильно, так как NULL не равен даже другому NULL.

Значение AUTOINCREMENT, как ограничение уровня столбца в базах данных SQLite. Ограничение AUTOINCREMENTв SQLite3.

AUTOINCREMENT является ограничением уровня столбца в базах данных, но обычно AUTOINCREMENT используется вместе с ограничением уровня таблицы PRIMARY KEY для создания суррогатных ключей. Давайте посмотрим, как AUTOINCREMENT поможет обеспечить целостность данных.

Создаемтаблицу:

|  |  |
| --- | --- |
| 1 | CREATE TABLE table5 (a INTEGER PRIMARY KEY AUTOINCREMENT, b TEXT NOT NULL); |

А теперь добавим данные в таблицу table5:

|  |  |
| --- | --- |
| 1 | INSERTINTO table5 (b) VALUES (‘какой-то текст’); |

Мы намеренно не стали указывать значение для первого столбца, чтобы посмотреть, как работает**ограничение столбца AUTOINCREMENT в базе данных SQLite3**. Сделаем выборку данных при помощи команды SELECT и посмотрим результат:

|  |  |
| --- | --- |
| 1  2  3  4  5 | -- ПроверяемработуограниченияAUTOINCREMENT    sqlite> select\* from table5;    1|какой-то текст |

Мы не добавляли единицу в первый столбец, за нас это сделало ограничение AUTOINCREMENT. Если вы создаете суррогатный ключ с ограничением уровня столбца AUTOINCREMNET, то не добавляйте вручную данные в ключевой столбец, СУБД будет делать это за вас.

**Вывод: ограничение AUTOINCREMENT используется для создания суррогатных ключей в таблицах базы данных.**

Значение CHECK, как ограничение уровня столбца в базах данных SQLite. Ограничение CHECKв SQLite3.

Мы упоминали о том, что CHECK является ограничением уровня столбца и используется CHECK для указания диапазона значений. Например, ограничение CHECK не даст записать значение в столбец меньше какого-то установленного минимума или больше какого-то максимума. Давайте посмотрим,**как работает ограничение столбца CHECK в базах данных SQLite**. Для этого создадим таблицу table6:

|  |  |
| --- | --- |
| 1 | CREATE TABLE table6 (a INTEGER PRIMARY KEY AUTOINCREMENT, b TEXT NOT NULL, c CHECK (c >95)); |

Мы создали таблицу с тремя столбцами и задали третьему столбцу ограничение столбца CHECK, теперь SQLite не даст нам возможность записать в столбец «c» значение равное 95 или меньше, обратите внимание: выражение в скобках может быть очень сложным и SQLite – это СУБД с динамической типизацией данных, поэтому будьте очень аккуратны, используя ограничение уровня столбца CHECK. Теперь давайте попробуем добавить данные:

|  |  |
| --- | --- |
| 1 | INSERT INTO table6 (b, c) VALUES (‘какой-тотекст’, 67); |

SQLite не даст нам возможность добавить эту строку, поскольку 67 меньше, чем 95. Мы получим предупреждение: Error: CHECK constraintfailed: table6. Оно говорит нам о том, что**сработало ограничение уровня столбца CHECK**.

Но, что будет, если мы попытаемся добавить значение NULL в столбец с ограничением CHECK? Давайте это выясним:

|  |  |
| --- | --- |
| 1 | INSERTINTOtable6 (b, c) VALUES (‘какой-тотекст’, NULL); |

SQLite корректно и честно добавит значение NULL в столбец с ограничением CHECK, это несколько странно, поскольку документация SQLite утверждает, что значение NULL всегда меньше любого другого значения при сравнении. Будьте аккуратны со значениями NULL при использовании ограничения уровня столбца CHECK.

**Вывод: ограничение уровня столбца CHECK в базах данных SQLiteпозволяет задать диапазон допустимых значений для хранимого значения в столбце.**

Мы рассмотрели ограничения уровня столбца, которые есть в базах данных SQLite все, кроме ограничения COLLATE, так как его мы подробно рассмотрели, когда говорили про сравнение данных в SQLite. Вам важно понимать, что **ограничения уровня столбца – это правила**, по которым работает база данных, причем правила за исполнение которых следит сама СУБД и не позволяет их нарушать. Ограничения уровня столбца позволяют нам не беспокоиться о значениях, которые хранятся в столбцах.