Базы данных

Определение данных, базовые операции с данными

SQLite version 3.31.1





**На этом уроке**

1. Узнаем, как изменять структуру базы данных.
2. Научимся добавлять, изменять и удалять данные.

**Оглавление**

[Изменение структуры базы данных](#_pu4yfvjsud2d)

[Изменение имени таблицы](#_vdn9odz8azdj)

[Изменение имени столбца](#_aqsx6epu2osy)

[Добавление столбца в таблицу](#_hcf2yeel0qei)

[Удаление таблицы](#_1tqm7cwrfx1r)

[Работа с данными](#_26yxg8qvv9m7)

[Простая выборка данных](#_kh8d0i4ro6wp)

[Добавление новых данных в таблицу](#_tvsjc27cxuc5)

[Вставка одной строки](#_pkom1o1hplf7)

[Вставка нескольких строк](#_e6xw4bmj30i3)

[Форматирование вывода данных](#_jwe8ktwhy85a)

[Режим вывода column](#_7ni1uws1089w)

[Режим вывода insert](#_ouc4uvgh03cm)

[Режим вывода line](#_6u4hiwgtw0nq)

[Режим вывода tabs](#_6bzglp9fytz9)

[Обновление существующих данных](#_6ncc4s9dpei2)

[Удаление данных](#_gqup62pa2iuu)

[Практическое задание](#_ftkawlaqqved)

[Глоссарий](#_tyjcwt)

[Дополнительные материалы](#_44sinio)

[Используемые источники](#_z337ya)

# Изменение структуры базы данных

Структура базы данных не считается предопределённой на весь срок жизни проекта. Она меняется, чтобы учесть новые требования. В СУБД SQLite для этого доступны следующие операции:

* изменение имени таблицы;
* изменение имени столбца;
* добавление нового столбца в таблицу.

## Изменение имени таблицы

Для внесения изменений используется команда ALTER TABLE.

Переименуем таблицу grades в final\_grades (итоговые оценки). Структурно команда выглядит следующим образом:

| ALTER TABLE 'Имя Таблицы' RENAME TO 'Новое Имя'; |
| --- |

Откроем базу данных students.db и посмотрим посредством команды .tables список существующих таблиц:

| sqlite> .tables courses grades streams students sqlite> .quit |
| --- |

Построим команду на переименование таблицы grades:

| ALTER TABLE grades RENAME TO final\_grades; |
| --- |

Выполним команду и проверим результат:

| sqlite> ALTER TABLE grades RENAME TO final\_grades;  sqlite> .tables  final\_grades courses streams students  sqlite> |
| --- |

## Изменение имени столбца

Переименуем столбец grade в final\_grade, для этого применим синтаксис RENAME COLUMN:

| ALTER TABLE 'Имя Таблицы' RENAME COLUMN 'Старое Имя Столбца' TO 'Новое Имя Столбца'; |
| --- |

Сейчас структура таблицы final\_grades выглядит следующим образом:

| sqlite> .schema final\_grades  CREATE TABLE IF NOT EXISTS "average\_grades" (  student\_id INTEGER NOT NULL,  stream\_id INTEGER NOT NULL,  grade REAL NOT NULL,  PRIMARY KEY(student\_id, stream\_id),  FOREIGN KEY (student\_id) REFERENCES students(id),  FOREIGN KEY (stream\_id) REFERENCES streams(id)  );  sqlite> |
| --- |

Подставим в команду имя таблицы, а также старое и новое имя столбца, которое надо изменить:

| ALTER TABLE final\_grades RENAME COLUMN grade TO final\_grade; |
| --- |

Выполним и проверим результат:

| sqlite> ALTER TABLE final\_grades RENAME COLUMN grade TO final\_grade;  sqlite> .schema final\_grades  CREATE TABLE IF NOT EXISTS "final\_grades" (  student\_id INTEGER NOT NULL,  stream\_id INTEGER NOT NULL,  final\_grade REAL NOT NULL,  PRIMARY KEY(student\_id, stream\_id),  FOREIGN KEY (student\_id) REFERENCES students(id),  FOREIGN KEY (stream\_id) REFERENCES streams(id)  );  sqlite> |
| --- |

## Добавление столбца в таблицу

Добавим в таблицу учеников адрес электронной почты (email) типа TEXT.

Посмотрим текущую структуру таблицы студентов:

| sqlite> .schema students  CREATE TABLE students (  id INTEGER PRIMARY KEY AUTOINCREMENT,  surname TEXT NOT NULL,  name TEXT NOT NULL  sqlite> |
| --- |

Команда для добавления нового столбца будет иметь следующий вид:

| ALTER TABLE students ADD COLUMN email TEXT; |
| --- |

Выполним команду и проверим результат:

| sqlite> ALTER TABLE students ADD COLUMN email TEXT;  sqlite> .schema students  CREATE TABLE students (  id INTEGER PRIMARY KEY AUTOINCREMENT,  surname TEXT NOT NULL,  name TEXT NOT NULL,  email TEXT);  sqlite> |
| --- |

## Удаление таблицы

Если таблица не требуется, то она удаляется. Но важно помнить, что таблица удаляется, только если на неё нет ссылок из других таблиц — нет внешних ключей других таблиц, ссылающихся на эту таблицу.

Для примера удалим таблицу оценок final\_grades:

| sqlite> .tables  final\_grades courses streams students  sqlite> |
| --- |

Для удаления таблиц применяется команда DROP TABLE:

| DROP TABLE 'Имя Таблицы'; |
| --- |

Удалим таблицу final\_grades:

| DROP TABLE final\_grades; |
| --- |

Проверим результат:

| sqlite> .tables  courses streams students  sqlite> |
| --- |

# Работа с данными

Мы уже умеем создавать структуру базы данных и вносить в неё корректировки. Но пустые базы данных бесполезны, поэтому сейчас разберём основные операции с данными:

* чтение (выборка) данных из таблицы;
* добавление (вставка) данных;
* изменение данных;
* удаление данных.

Для сокращённого определения этих действий часто применяется обозначение CRUD — от английских слов Create, Read, Update и Delete.

Начнём работу с таблицы курсов, данные которой выглядят следующим образом:

| **Ключ курса** | **Название курса** | **Количество уроков** |
| --- | --- | --- |
| 1 | Базы данных | 12 |
| 2 | Linux. Рабочая станция | 8 |
| 3 | Основы Python | 8 |

*Таблица 4. Таблица курсов*

Вспомним также структуру этой таблицы:

| sqlite> .schema courses  CREATE TABLE courses (  id INTEGER PRIMARY KEY AUTOINCREMENT,  name TEXT NOT NULL UNIQUE,  lessons\_amount INTEGER  );  sqlite> |
| --- |

## Простая выборка данных

Прежде чем работать с данными таблицы, убедимся, что она пустая. Для этого понадобится команда SELECT. В базовом варианте для этой команды достаточно указать, к какой таблице мы обращаемся, и перечислить имена столбцов, данные которых нас интересуют:

| SELECT 'Столбец 1', 'Столбец 2', 'Столбец N' FROM 'Имя Таблицы'; |
| --- |

Если надо получить данные всех столбцов, то применяется символ подстановки \* (звёздочка). Это упростит команду:

| SELECT \* FROM 'Имя Таблицы'; |
| --- |

В соответствии со стандартом языка запросов SQL идентификаторы — имена таблиц и столбцов — заключаются в двойные кавычки, а строковые значения выделяются одинарными кавычками. В SQLite для совместимости с другими СУБД используются одинарные кавычки для определения идентификаторов и двойные — для строк. Обратные кавычки применяются для идентификаторов. Все команды ниже выполнятся одинаковым образом:

| SELECT \* FROM "courses"; SELECT \* FROM 'courses'; SELECT \* FROM `courses`; |
| --- |

На практике в различных проектах встречаются разные стили использования кавычек. Следует придерживаться того стиля, который уже используется в коде, чтобы проект выглядел целостным.

Часто названия таблиц и столбцов вообще остаются без кавычек. Это допускается, если имена задаются латиницей и не содержат пробелы, спецсимволы и зарезервированные слова, которые используются в синтаксисе SQL. Строковые значения данных при этом обязательно заключаются в двойные или одинарные кавычки. В примерах, рассмотренных в этом курсе, предлагается применить именно такой стиль использования кавычек, поэтому команду выборки информации лучше упростить:

| SELECT \* FROM courses; |
| --- |

Ожидаемо мы получим пустую выборку, но теперь у нас есть инструмент, чтобы контролировать другие операции работы с данными.

| sqlite> SELECT \* FROM courses;  sqlite> |
| --- |

## Добавление новых данных в таблицу

Следующий шаг — использование данных. Очевидно, что сначала надо поместить данные в наши таблицы, чтобы иметь возможность работать с ними далее.

Для добавления данных (вставка строк таблицы) применяется команда INSERT, куда указывается имя таблицы, список столбцов, в которые добавляются данные и, конечно, сами данные:

| INSERT INTO 'Имя Таблицы' [('Столбец 1', 'Столбец 2', 'Столбец 3', 'Столбец N')]  VALUES ('Значение 1', 'Значение 2', 'Значение 3', 'Значение N'); |
| --- |

### Вставка одной строки

Рассмотрим, как вставляется одна строка в таблицу курсов:

| INSERT INTO courses (id, name, lessons\_amount) VALUES (1, 'Базы данных', 12); |
| --- |

Обратите внимание, что строковые значения мы записываем в кавычках.

Если данные вставляются во все столбцы таблицы, то в команде INSERT имена столбцов не перечисляются. В этом случае команда выглядит так:

| INSERT INTO courses VALUES (1, 'Базы данных', 12); |
| --- |

Вспомним, что столбец первичного ключа id имеет признак автоинкремента (AUTOINCREMENT), поэтому вставлять значение для id явным образом необязательно:

| INSERT INTO courses (name, lessons\_amount) VALUES ('Базы данных', 12); |
| --- |

Выполним последний вариант команды и проверим результат выборкой SELECT:

| sqlite> INSERT INTO courses (name, lessons\_amount) VALUES ('Базы данных', 12);  sqlite> SELECT \* FROM courses;  1|Базы данных|12  sqlite> |
| --- |

Мы видим, что данные успешно добавлены в таблицу.

### Вставка нескольких строк

Есть возможность вставлять сразу несколько строк. В этом случае перечислим добавляемые данные для каждой строки через запятую. Добавим остальные данные таким способом:

| INSERT INTO courses (name, lessons\_amount) VALUES  ('Linux. Рабочая станция', 8),  ('Основы Python', 8); |
| --- |

Обратите внимание, что мы перенесли эту команду на несколько строк. Это возможно и даже рекомендуется делать, чтобы код команды выглядел структурированным и его было проще анализировать. Нет единственно верного способа форматировать код команд SQL, но есть рекомендации, как это сделать. Пример таких рекомендаций — [здесь](https://www.sqlstyle.guide/ru/)**.**

Выбор окончательного варианта оформления кода — за вами. Но если вы начинаете работать над уже существующим проектом, то требуется придерживаться уже применяемого стиля. Стремитесь к тому, чтобы ваш код SQL выглядел понятным и структурированным.

Выполним команду и проверим результат:

| sqlite> INSERT INTO courses (name, lessons\_amount) VALUES  ...> ('Linux. Рабочая станция', 8),  ...> ('Основы Python', 8);  sqlite> SELECT \* FROM courses;  1|Базы данных|12  2|Linux. Рабочая станция|8  3|Основы Python|8  sqlite> |
| --- |

## Форматирование вывода данных

В клиенте sqlite3 задаются различные форматы для вывода результатов запросов.

Чтобы посмотреть текущий формат вывода, применяется команда .mode:

| sqlite> .mode  current output mode: list  sqlite> |
| --- |

Как видим, режим вывода по умолчанию — list. В этом режиме данные выводятся построчно и разделяются вертикальной чертой:

| sqlite> .mode  current output mode: list  sqlite> SELECT \* FROM courses;  1|Базы данных|12  2|Linux. Рабочая станция|8  3|Основы Python|8  sqlite> |
| --- |

Рассмотрим другие режимы, полезные в рамках курса:

* column — табличный вывод, строки обрезаются;
* insert — вывод команд вставки INSERT;
* line — вывод названий столбцов с соответствующими значениями (вертикальный вывод), строки не обрезаются;
* tabs — табличный вывод, строки не обрезаются.

### Режим вывода column

В режиме column данные выводятся по столбцам, а длинные строки обрезаются по правому краю:

| sqlite> .mode column  sqlite> SELECT \* FROM courses;  1 Базы данных 12  2 Linux. Рабо 8  3 Основы Pyth 8  sqlite> |
| --- |

### Режим вывода insert

Режим insert покажет команды вставки, которые надо выполнить для создания данных выборки:

| sqlite> .mode insert  sqlite> SELECT \* FROM courses;  INSERT INTO "table" VALUES(1,'Базы данных',12);  INSERT INTO "table" VALUES(2,'Linux. Рабочая станция',8);  INSERT INTO "table" VALUES(3,'Основы Python',8);  sqlite> |
| --- |

### Режим вывода line

В режиме line для каждой записи выводятся наименования столбцов и соответствующие значения для строк. Такой режим удобно применять, чтобы получить длинные строковые значения без обрезки:

| sqlite> .mode line  sqlite> SELECT \* FROM courses;  id = 1  name = Базы данных  lessons\_amount = 12  id = 2  name = Linux. Рабочая станция  lessons\_amount = 8  id = 3  name = Основы Python  lessons\_amount = 8  sqlite> |
| --- |

### Режим вывода tabs

Режим tabs позволяет вывести данные в табличном виде без обрезания строк:

| sqlite> .mode tabs  sqlite> SELECT \* FROM courses;  1 Базы данных 12  2 Linux. Рабочая станция 8  3 Основы Python 8  sqlite> |
| --- |

На курсе мы применим различные режимы вывода в зависимости от задачи и характера данных, но в основном будем использовать режим column с включённым отображением имён столбцов:

| sqlite> .header on  sqlite> .mode column  sqlite> SELECT \* FROM courses;  id name lessons\_amount  ---------- ----------- --------------  1 Базы данных 12  2 Linux. Рабо 8  3 Основы Pyth 8  sqlite> |
| --- |

## Обновление существующих данных

Чтобы обновить существующие данные, используется команда UPDATE. В команде мы указываем имя таблицы, передаём список изменяемых значений, а также определяем условие, которое указывает, какая именно строка (или строки) требует изменений:

| UPDATE 'Имя Таблицы' SET 'Столбец 1' = 'Значение1', 'Столбец 2' = 'Значение 2', 'Столбец N' = 'Значение N' WHERE ['Условие']; |
| --- |

Для примера изменим количество уроков курса «Базы данных» и укажем соответствующий идентификатор строки:

| UPDATE courses SET lessons\_amount = 8 WHERE id = 1; |
| --- |

Выполним команду и проверим результат:

| sqlite> UPDATE courses SET lessons\_amount = 8 WHERE id = 1;  id name lessons\_amount  ---------- ----------- --------------  1 Базы данных 12  2 Linux. Рабо 8  3 Основы Pyth 8  sqlite> |
| --- |

Другим вариантом обновления этой строки будет определение условия по названию курса:

| UPDATE courses SET lessons\_amount = 8 WHERE name = 'Базы данных'; |
| --- |

Если мы можем получить результат двумя вариантами, то какой способ предпочтителен? Когда мы вводим в условие поиска название курса, то есть вероятность ошибиться в написании или перепутать с другим похожим названием. Если используем идентификатор строки для этой цели, то верный результат гарантирован. Такой подход рекомендуется в тех случаях, когда идентификатор записи уже известен.

## Удаление данных

Для удаления данных используется команда DELETE, в которой задаём имя таблицы, а также условие для удаления конкретных строк:

| DELETE FROM 'Имя таблицы' WHERE [Условие]; |
| --- |

**Важно!** *Если для команды DELETE не указать условие, то удалятся все строки таблицы.*

Команда удаления курса с именем 'Основы Python':

| DELETE FROM courses WHERE name = 'Основы Python'; |
| --- |

Выполним команду и проверим результат:

| sqlite> DELETE FROM courses WHERE name = 'Основы Python';  sqlite> SELECT \* FROM courses;  id name lessons\_amount  ---------- ----------- --------------  1 Базы данных 12  2 Linux. Рабо 8  sqlite> |
| --- |

# Практическое задание

Работаем с базой данных teachers.db. В качестве отчёта сдайте команды, которые выполнялись (в текстовом файле), а также файл базы данных.

1. В таблице streams переименуйте столбец даты начала обучения в started\_at.
2. В таблице streams добавьте столбец даты завершения обучения в finished\_at.
3. Приведите данные в полное соответствие с данными в таблицах ниже:

| **id** | **name** | **surname** | **email** |
| --- | --- | --- | --- |
| 1 | Николай | Савельев | saveliev.n@mai.ru |
| 2 | Наталья | Петрова | petrova.n@yandex.ru |
| 3 | Елена | Малышева | malisheva.e@google.com |

*Таблица 1. Преподаватели (teachers)*

| **id** | **name** |
| --- | --- |
| 1 | Базы данных |
| 2 | Основы Python |
| 3 | Linux. Рабочая станция |

*Таблица 2. Курсы (courses)*

| **id** | **course\_id** | **number** | **started\_at** | **students\_amount** |
| --- | --- | --- | --- | --- |
| 1 | 3 | 165 | 18.08.2020 | 34 |
| 2 | 2 | 178 | 02.10.2020 | 37 |
| 3 | 1 | 203 | 12.11.2020 | 35 |
| 4 | 1 | 210 | 03.12.2020 | 41 |

*Таблица 3. Потоки (streams)*

| **teacher\_id** | **stream\_id** | **performance** |
| --- | --- | --- |
| 3 | 1 | 4.7 |
| 2 | 2 | 4.9 |
| 1 | 3 | 4.8 |
| 1 | 4 | 4.9 |

*Таблица 4. Успеваемость (academic\_performance)*

1. Дополнительное задание (выполняется по желанию): в таблице успеваемости измените тип столбца «Ключ потока» на REAL. Выполните задание на таблице с данными.

# Глоссарий

**CRUD** — основные операции для работы с данными, сокращение от английского Create, Read, Update и Delete, соответственно — создание, чтение, обновление и удаление данных.

# Дополнительные материалы

1. [Документация SQLite, изменение таблицы](https://unetway.com/tutorial/sqlite-alter-table).
2. [Документация SQLite, выборка данных](https://unetway.com/tutorial/sqlite-command-select).
3. [Документация SQLite, вставка данных](https://unetway.com/tutorial/sqlite-command-insert).
4. [Документация SQLite, обновление данных](https://unetway.com/tutorial/sqlite-operator-update).
5. [Документация SQLite, удаление данных](https://unetway.com/tutorial/sqlite-operator-delete).
6. [Руководство по стилю SQL](https://www.sqlstyle.guide/ru/).

# Используемые источники

1. [Документация SQLite, изменение таблицы](https://www.sqlite.org/lang_altertable.html).
2. [Документация SQLite, выборка данных](https://www.sqlite.org/lang_select.html).
3. [Документация SQLite, вставка данных](https://www.sqlite.org/lang_insert.html).
4. [Документация SQLite, обновление данных](https://www.sqlite.org/lang_update.html).
5. [Документация SQLite, удаление данных](https://www.sqlite.org/lang_delete.html).