GeekBrains

Базы данных

Базовые понятия структуры базы данных

SQLite version 3.31.1

На этом уроке

1. Научимся создавать базы данных в СУБД SQLite.
2. Узнаем, как определять связи между таблицами.
3. Попрактикуемся создавать таблицы баз данных.

Оглавление

Дополнительные материалы

Используемые источники

Основные этапы по созданию таблиц базы данных

Мы знаем, когда формируются структуры реляционной базы данных, сущности реального мира отображаются на таблицы, атрибуты сущности — на столбцы таблицы, а данные экземпляров сущности — на строки таблицы. Проще говоря, наименованиями столбцов таблицы будут имена атрибутов учеников (имя, фамилия), а в строках запишутся данные каждого учащегося.

Команда CREATE TABLE

Чтобы создать таблицу, надо воспользоваться командой CREATE TABLE, в которой потребуется указать имя таблицы и список столбцов через запятую в скобках.

```
CREATE TABLE ИмяТаблицы(
  Столбец 1 Тип Ограничения Первичный Ключ,
  Столбец 2 Тип Ограничения,
  Столбец 3 Тип Ограничения,
  .....
  Столбец N Тип Ограничения
);
```

Важно! *Ввод команд языка SQL должен завершаться точкой с запятой.*

Чтобы правильно использовать эту команду и понять, какие параметры нам надо задать в том или ином случае, определим алгоритм действий, который затем применим для создания таблиц базы данных учеников.

Выбираем имя таблицы

В качестве имени таблицы подходящим вариантом будет имя сущности во множественном числе, например, «Ученики», «Курсы», «Потоки», «Оценки». Множественное число применяем потому, что в таблицах хранятся данные о многих учениках, курсах, потоках и оценках.

Определяем список столбцов и их тип

В столбцах хранятся различные данные: имя ученика — это строка, ключ — целое число, начало занятий задаётся датой. При создании таблицы надо определить подходящие типы данных для каждого столбца.

В СУБД SQLite используются пять типов данных.

Данные	Тип данных	Описание
Нет значения	NULL	NULL — «ничего». Это непустая строка или ноль, NULL — отсутствие значения.
Целое число	INTEGER	Целое число со знаком, например, 1, 56, -764.
Дробное число	REAL	Число с плавающей точкой, например, 34.568, -4.2
Строка, текст	TEXT	Строка текста, например, «Базы данных»
Двоичные данные, медиа	BLOB	Этот тип предназначен для хранения данных в исходном формате. В столбце такого типа сохраняется, например, изображение.

Таблица 1. Типы данных SQLite

Определяем дополнительные условия для столбцов

В соответствии с логикой приложения часто требуется указать дополнительные условия для некоторых столбцов. Например, сохранять данные об ученике без его фамилии имеет мало смысла. Поэтому при создании таблицы определяется специальное условие, разрешающее сохранение только непустого значения фамилии, оно выдаёт ошибку, если предпринимается попытка вставить строку без фамилии ученика. Такие условия называются ограничениями, так как мы ограничиваем использование данных в столбце только некоторыми допустимыми значениями.

Общепринятое название	Синтаксис SQL	Назначение ограничения
Ограничение на пустое (NULL) значение	NOT NULL	Гарантирует, что в ячейку вставится непустое значение
Ограничение уникальности	UNIQUE	Гарантирует уникальность значений в пределах таблицы
Значение по умолчанию	DEFAULT	Определяет значение по умолчанию для столбца, если оно не задано явным образом

Таблица 2. Ограничения данных

Определяем ключевые столбцы

Для каждой таблицы требуется ключевой столбец. Термин «ключевой» означает, что значения этого столбца уникальны в пределах таблицы. Значения в ключевых столбцах представляют собой идентификаторы, по которым осуществляется поиск подходящей строки таблицы.

Ключевой столбец называется также столбцом первичного ключа, или просто первичным ключом. Значения в ячейках этого столбца — идентификаторы строк.

Синтаксически первичный ключ определяется как PRIMARY KEY. Чаще всего значения первичного ключа определяются условными целочисленными значениями — 1.2.3 и так далее. Чтобы пользователям не потребовалось следить за нумерацией и исключать появление ошибок при вставке значений первичного ключа, используется специальный признак автоматического приращения значения AUTOINCREMENT.

Общепринятое название	**Синтаксис SQL**	**Назначение ограничения**
Первичный ключ	PRIMARY KEY	Определяет первичный ключ таблицы
Автоинкремент	AUTOINCREMENT	При добавлении новой строки СУБД автоматически подставит следующее значение ключа, равное значению ключа последней строки, прибавив единицу.

Таблица 3. Ограничение первичного ключа и автоприращение

Определяем связи между таблицами

Чтобы разобраться с различными вариантами связи между таблицами базы данных, рассмотрим примеры отношений реального мира.

Отношение «один к одному»

У каждого совершеннолетнего ученика есть паспорт, но не может быть нескольких действующих паспортов одновременно — имеется в виду гражданство одной страны. В то же время этот паспорт не принадлежит нескольким ученикам. В этом случае мы получим связь «один к одному» между учеником и его паспортом.

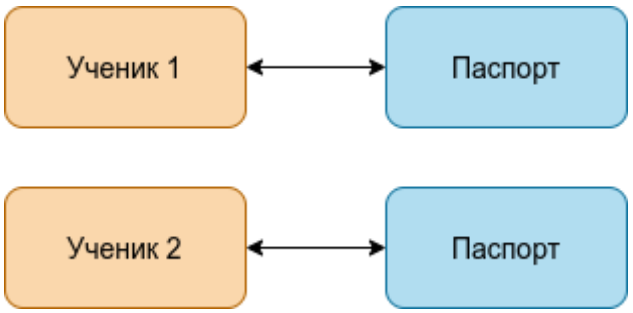

Диаграмма 1. Отношение «один к одному»

Связь «один к одному» — каждому экземпляру сущности А соответствует один и только один экземпляр сущности Б и, наоборот, каждому экземпляру сущности Б соответствует один и только один экземпляр сущности А.

Отношение «один ко многим»

Например, ученик владеет несколькими книгами, при этом каждая книга принадлежит только одному ученику. В этом случае мы видим связь «один ко многим».

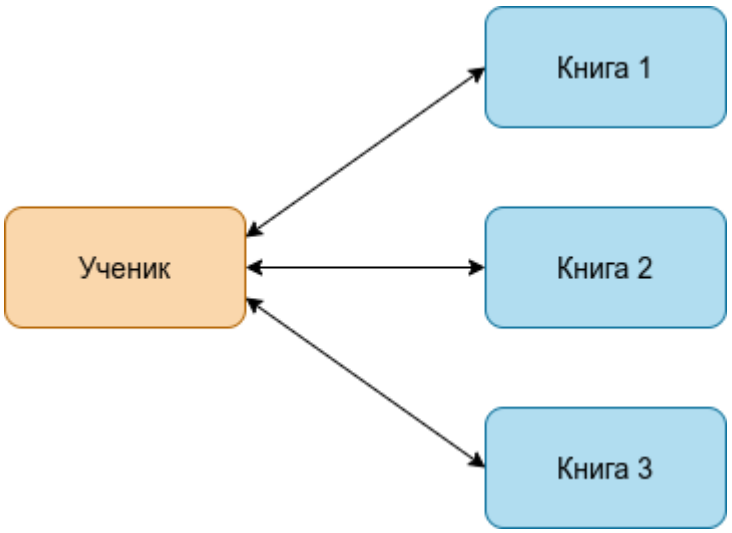

Диаграмма 2. Отношение «один ко многим»

Связь «один ко многим» — одному экземпляру сущности А соответствует несколько экземпляров сущности Б, однако каждому экземпляру сущности Б соответствует только один экземпляр сущности А.

Отношение «многие ко многим»

Когда ученик регистрируется в нескольких социальных сетях, то мы получим другой вариант отношения. В этом случае ученик — участник нескольких сетей, но эти социальные сети, в свою очередь, включают множество пользователей. Такой тип отношения называется «многие ко многим».

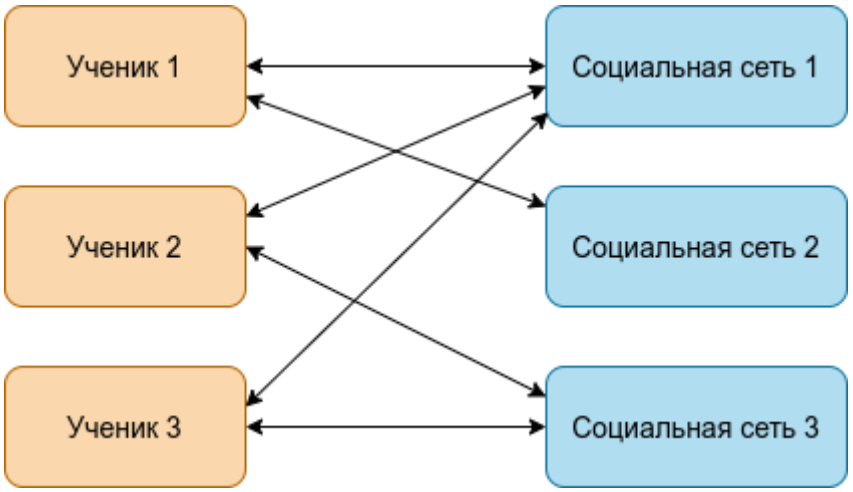

Диаграмма 3. Отношение «многие ко многим»

Связь «многие ко многим» — каждому экземпляру сущности А соответствует несколько экземпляров сущности Б, и наоборот.

Отношения между таблицами в базе данных учеников

Вспомним диаграмму связи между таблицами, которую мы рассмотрели на первом занятии.

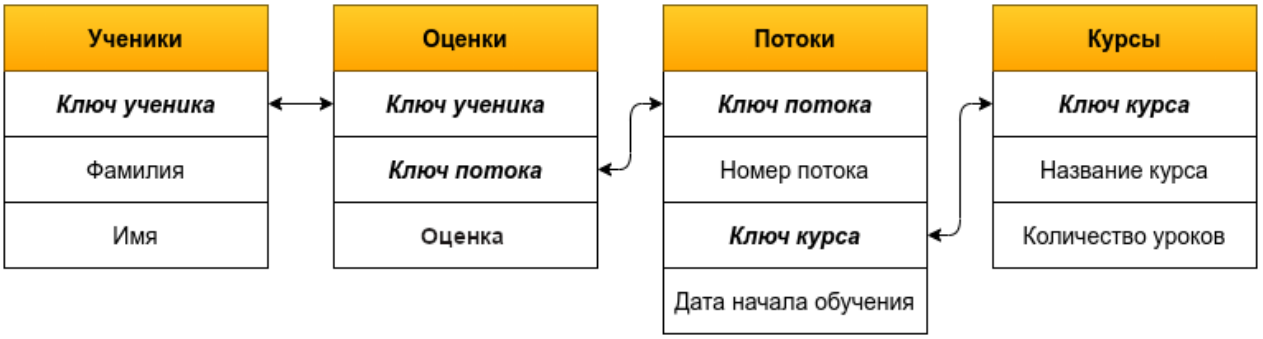

Диаграмма 4. Связи между таблицами

Связь «ученики — оценки»

Ученик получает одну оценку за курс, но курсов может быть много. Следовательно, оценок у ученика окажется столько же. В то же время каждая оценка отражает работу одного конкретного ученика при прохождении одного курса. То есть оценка не принадлежит нескольким ученикам. В этом случае мы видим связь «один ко многим»: один ученик — много оценок.

Связь «оценки — потоки»

Оценка определяет успеваемость студента на конкретном потоке (курсе). Эта оценка не относится к нескольким различным потокам. В то же время в потоке много учеников, поэтому для одного потока

предусматривается много оценок (по количеству учеников). В этом случае мы также видим связь «один ко многим»: один поток — много оценок.

Связь «ученики — потоки»

Интересно рассмотреть связь между учениками и потоками, они связаны через таблицу оценок. Например, один ученик проходит обучение в нескольких потоках. В то же время в одном потоке обучается некоторое количество учеников. В этом случае мы видим связь «многие ко многим»: много учеников — много потоков.

Связь «потоки — курсы»

Например, по одному курсу проводится много потоков, при этом каждый поток относится только к одному курсу. Потоки и курсы связаны отношением «один ко многим»: один курс — много потоков.

Внешние ключи

Выше мы определили связи между таблицами, но пока только на логическом уровне. Чтобы задать связи между таблицами на уровне базы данных, надо определить эти отношения формальным образом. Определяются так называемые внешние ключи.

Столбцом внешнего ключа, или просто внешним ключом, называется столбец, значения которого ссылаются на значения столбца первичного ключа в другой таблице. Например, в таблице оценок есть столбец «Ключ ученика», где значения этого столбца ссылаются на значения столбца первичного ключа «Ключ ученика» в таблице учеников.

Для таких столбцов ограничение внешнего ключа определяется посредством выражения FOREIGN KEY, в котором также надо указать, куда ссылается внешний ключ.

В нашем случае это выражение выглядит следующим образом:

```
FOREIGN KEY (КлючУченика) REFERENCES Ученики(КлючУченика)
```

Это выражение дословно значит следующее:

```
ВНЕШНИЙ КЛЮЧ (КлючУченика) ССЫЛАЕТСЯ НА Ученики(КлючУченика)
```

Создаём таблицы базы данных

На этом занятии наша задача — построить базы данных студентов в СУБД SQLite3 на основе таблиц, которые мы получили в результате нормализации данных на первом занятии.

Эта структура соответствует требованиям реляционной модели данных, и теперь мы можем создать аналогичные таблицы в базе данных students.db посредством команд языка SQL.

В целом, процесс создания таблицы сводится к следующим последовательным шагам:

1. Выбираем имя таблицы.
2. Определяем список столбцов и их тип.
3. Определяем дополнительные условия для столбцов.
4. Определяем первичный ключ таблицы.
5. Определяем внешние ключи.
6. Создаем таблицу командой CREATE TABLE.
7. Проверяем результат.

База данных учеников

Запустим программу sqlite3. Способ запуска зависит от варианта установки. Варианты для различных десктопных операционных систем мы рассмотрели на первом занятии.

После запуска клиента появятся некоторые сообщения, а также приглашение для ввода команд.

```
SQLite version 3.31.1 2020-01-27 19:55:54
Enter ".help" for usage hints.
Connected to a transient in-memory database.
Use ".open FILENAME" to reopen on a persistent database.
sqlite>
```

Нам потребуется создать новую базу данных учеников. Для этого воспользуемся командой .open (точка + open), которая открывает действующую базу данных или создаёт новую, если базы данных с таким именем не найдено.

```
sqlite>.open students.db
sqlite>
```

Мы создали базу данных. Одноимённый файл базы данных students.db теперь находится в той директории, откуда запустился клиент sqlite3.

Важно! *После того как база данных сформировалась, надо запустить клиент sqlite3 из той папки, где находится файл students.db. Иначе появится новая пустая база данных с тем же именем, но в другой папке.*

Команды с точкой (дот-команды)

Вероятно, вы обратили внимание, что мы открыли базу данных с использованием команды с точкой. Такие команды не считаются командами языка запросов SQL и используются только в СУБД SQLite. Другое название таких команд — дот-команды (dot — точка).

Таблица курсов

Начнём с таблицы курсов. Исходная структура выглядит 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

Очевидное и подходящее имя для таблицы базы данных — «Курсы». В большинстве современных проектов для именования таблиц и столбцов используются слова английского языка. Это связано с тем, что компании и проекты часто интернациональны, заказчики также могут быть из разных стран. Имя таблицы курсов определим как courses.

Важно! *Применять транслит (например, kursi) нельзя, так как это выглядит непрофессионально.*

Проанализируем таблицу и определим тип данных и набор ограничений для каждого столбца.

Для ключевого столбца подойдёт тип INTEGER, для него также надо задать ограничение первичного ключа PRIMARY KEY и признак автоинкремента AUTOINCREMENT. А для столбца первичного ключа используется имя id (identifier — идентификатор).

Столбцу «Название курса» используем тип TEXT. Так как запись о курсе без названия не имеет смысла, добавим ограничение на ввод пустых значений NOT NULL. Названия курсов не должны повторяться, поэтому включим сюда ограничение уникальности.

Для столбца «Количество уроков» подойдёт тип данных INTEGER.

Сведём типы данных и ограничения по столбцам в одну таблицу.

Имя столбца	Тип данных	Ограничения

id	INTEGER	PRIMARY KEY AUTOINCREMENT
name	TEXT	NOT NULL UNIQUE
lessons_amount	INTEGER	

Таблица 5. Столбцы таблицы courses

Теперь у нас есть все данные, чтобы скомпоновать команду CREATE TABLE.

```
CREATE TABLE courses (
  id INTEGER PRIMARY KEY AUTOINCREMENT,
  name TEXT NOT NULL UNIQUE,
  lessons_amount INTEGER
);
```

Результат выполнения команды выглядит следующим образом:

```
sqlite> CREATE TABLE courses (
   ...>   id INTEGER PRIMARY KEY AUTOINCREMENT,
   ...>   name TEXT NOT NULL UNIQUE,
   ...>   lessons_amount INTEGER
   ...> );
sqlite>
```

Проверим, что мы действительно создали таблицу:

```
sqlite> .tables
courses
sqlite>
```

Чтобы получить более полную информацию о таблице, используем команду .schema:

```
sqlite> .schema courses
CREATE TABLE courses (
  id INTEGER PRIMARY KEY AUTOINCREMENT,
  name TEXT NOT NULL UNIQUE,
  lessons_amount INTEGER
);
sqlite>
```

Таблица потоков

Анализируем таблицу потоков

Ключ потока	Номер потока	Ключ курса	Дата начала обучения
1	45	2	18.08.2020
2	48	3	02.10.2020
3	54	1	12.11.2020

Таблица 6. Таблица потоков

Определим типы данных и ограничения для столбцов.

Имя столбца	Тип данных	Ограничения
id	INTEGER	PRIMARY KEY AUTOINCREMENT
number	INTEGER	NOT NULL UNIQUE
course_id	INTEGER	NOT NULL FOREIGN KEY
start_date	TEXT	NOT NULL

Таблица 7. Столбцы таблицы streams

Обратите внимание, что для столбца «Начало обучения» мы используем тип данных TEXT, так как в SQLite нет отдельного типа данных для даты.

В этой таблице также есть столбец внешнего ключа «Ключ курса», значения которого ссылаются на значения первичного ключа «Ключ курса» в таблице курсов.

Столбцы внешнего ключа принято именовать специальным образом — имя сущности, на которую ссылается внешний ключ, плюс суффикс _id. В нашем случае это course_id, и, увидев такое имя столбца, мы сразу понимаем, что значения этого столбца ссылаются на столбец id таблицы курсов courses.

Команда создания таблицы выглядит следующим образом:

```
CREATE TABLE streams (
  id INTEGER PRIMARY KEY AUTOINCREMENT,
  number INTEGER NOT NULL UNIQUE,
  course_id INTEGER NOT NULL,
  start_date TEXT NOT NULL,
  FOREIGN KEY (course_id) REFERENCES courses(id)
);
```

Таблица учеников

Таблица учеников структурно выглядит несложно.

Ключ ученика	Фамилия	Имя
1	Иванов	Игорь
2	Павлова	Анастасия

Таблица 8. Таблица учеников

Имя столбца	Тип данных	Ограничения
id	INTEGER	PRIMARY KEY AUTOINCREMENT
surname	TEXT	NOT NULL
name	TEXT	NOT NULL

Таблица 9. Столбцы таблицы students

Команда создания таблицы students:

```
CREATE TABLE students (
  id INTEGER PRIMARY KEY AUTOINCREMENT,
  surname TEXT NOT NULL,
  name TEXT NOT NULL
);
```

Таблица оценок

Таблица оценок включает в себя ссылки на учеников и потоки, а также оценку за курс.

Ключ ученика	Ключ потока	Оценка
1	1	4.9
2	1	5
1	3	5
1	2	4.9

Таблица 10. Таблица оценок

Имя столбца	Тип данных	Ограничения

student_id	INTEGER	NOT NULL PRIMARY KEY FOREIGN KEY
stream_id	INTEGER	NOT NULL PRIMARY KEY FOREIGN KEY
grade	REAL	NOT NULL

Таблица 11. Столбцы таблицы grades

Обратите внимание, что в таблице оценок нет ключевого столбца «Ключ оценки». Вместо этого, мы определили ограничение первичного ключа сразу на два столбца — «Ключ ученика» и «Ключ потока». Дело в том, что ученик получает только одну среднюю оценку по всем практическим заданиям за курс (поток). Поэтому комбинация значений ключа ученика и ключа потока уникальна в пределах всей таблицы.

В этом случае применяем составной первичный ключ, который состоит из комбинации значений нескольких столбцов.

Столбец «Ключ ученика» ссылается на таблицу учеников, а столбец «Ключ потока» — на таблицу потоков. Оба этих столбца — кандидаты на определение в качестве внешних ключей.

Команда создания таблицы grades:

```
CREATE TABLE 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)
);
```

Составной первичный ключ задается отдельным выражением `PRIMARY KEY(student_id, stream_id)`.

Проверим, что все таблицы успешно созданы, и выйдем из sqlite3.

```
sqlite> .tables
courses      grades     streams    students
sqlite> .quit
```

Практическое задание

1. Создайте в SQLite базу данных преподавателей и назовите её teachers.db. В этой базе данных сформируйте таблицы преподавателей (teachers), курсов (courses), потоков (streams) и

успеваемости (grades) на основе структур, которые представлены ниже. Обратите внимание, что данные вводить пока не надо. Сдайте отчет в виде файла базы данных teachers.db.

Таблица 1. Преподаватели

Ключ преподавателя	**Имя**	**Фамилия**	**Электронная почта**
1	Николай	Савельев	saveliev.n@mai.ru
2	Наталья	Петрова	petrova.n@yandex.ru
3	Елена	Малышева	malisheva.e@google.com
4	Никита	Ващенко	vashenko.n@yandex.ru

Таблица 2. Курсы

Ключ курса	**Название**
1	Базы данных
2	Основы Python
3	Linux. Рабочая станция

Таблица 3. Потоки

Ключ потока	**Ключ курса**	**Номер потока**	**Дата начала обучения**	**Количество учеников**
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 4. Успеваемость

Ключ преподавателя	**Ключ потока**	**Успеваемость**
3	1	4.7
2	2	4.9
1	3	4.8
1	4	4.9

2. Дополнительное задание (выполняется по желанию): определите все связи между таблицами, созданными по результатам первого задания. Укажите, какой тип связи используется в каждом случае.

Глоссарий

Ограничение — правило для столбца таблицы, которое определяет допустимость вводимых значений.

Первичный ключ, столбец первичного ключа — столбец, значения которого уникальны в пределах таблицы и используются для однозначной идентификации строк в пределах таблицы.

Составной первичный ключ — ключ, который состоит из комбинации значений нескольких столбцов.

Автоинкремент — дополнительный признак столбца, при использовании которого СУБД самостоятельно генерирует значение, прибавляя единицу к предыдущему значению.

Один к одному — тип связи, когда каждому экземпляру сущности А соответствует один и только один экземпляр сущности Б и, наоборот, каждому экземпляру сущности Б соответствует один и только один экземпляр сущности А.

Один ко многим — тип связи, когда одному экземпляру сущности А соответствует несколько экземпляров сущности Б, однако каждому экземпляру сущности Б соответствует только один экземпляр сущности А.

Многие ко многим — тип связи, при котором каждому экземпляру сущности А соответствует несколько экземпляров сущности Б, и наоборот.

Внешний ключ, столбец внешнего ключа — столбец, значения которого ссылаются на значения столбца первичного ключа в другой таблице.

Дот-команды — команды СУБД SQLite, которые начинаются с точки. Эти команды не считаются командами языка SQL.

Дополнительные материалы

1. Документация SQLite, команды с точкой.
2. Документация SQLite, создание таблицы.
3. Статья «Условия в SQL».

Используемые источники

1. SQLite, документация на русском языке.

2. Статья «Взаимосвязи и модели».