Создание таблиц

К этому моменту мы познакомились с четырьмя главными запросами языка SQL, которые предназначены для работы с данными:

- SELECT – для извлечения данных
- INSERT – для добавления данных
- UPDATE – для изменения данных
- DELETE – для удаления данных

Мы изучили далеко не все возможности этих запросов, и продолжим их изучение в следующих уроках этого курса. А пока давайте зададимся вопросом: а как создаются сами таблицы? Как можно их изменять и уничтожать?

Для этого тоже используется язык SQL. В него включены особые запросы, предназначенные для создания, изменения и удаления объектов баз данных, в том числе (и в первую очередь) таблиц. Поскольку они не предназначены для работы с данными, часто их называют не запросами, а командами языка SQL, и мы тоже будем их так называть. Для этих команд намеренно выбраны ключевые слова, не совпадающие с запросами для работы с данными:

- CREATE – для создания объектов БД
- ALTER – для изменения объектов БД
- DROP – для удаления объектов БД

После этих ключевых слов всегда добавляется ещё одно слово, обозначающее тип объекта. Для таблиц это ключевое слово TABLE. Соответственно, команды SQL, предназначенные для создания, изменения и удаления таблиц начинаются со слов:

- CREATE TABLE
- ALTER TABLE
- DROP TABLE

Позже в этом курсе мы научимся работать с некоторыми другими типами объектов баз данных, а сейчас давайте познакомимся поближе с командой CREATE TABLE.

В обобщённом виде структуру простейшей команды CREATE TABLE можно представить так:

```
create table {имя таблицы} ({описание полей})
```

Например, так может выглядеть команда для создания таблицы Categories, которая присутствует в обеих наших учебных базах данных:

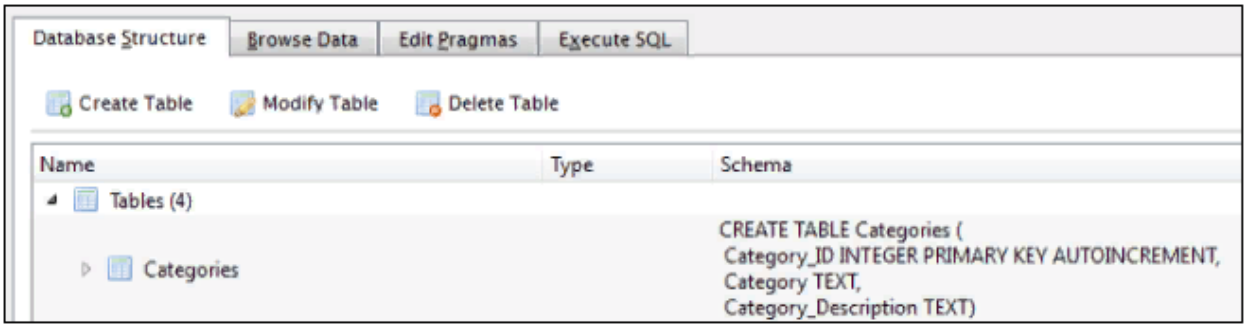
```
create table Categories
(Category_ID INTEGER,
Category TEXT,
Category_Description TEXT)
```

При выполнении эта команда создаст таблицу, структура которой включает три поля. Описание каждого поля включает его имя и тип. Например, следующее описание содержит имя поля **Category** и тип поля **TEXT:**

```
Category TEXT
```

В списке описания полей разделяются запятыми.

Приведенный пример несколько упрощён. Вы можете сравнить эту команду с той, которую показывает программа SQLite Database Browser на вкладке Database Structure:



Вы видите на этой иллюстрации некоторые элементы, которых нет в нашей команде: в описании поля Category_ID присутствуют незнакомые нам пока ключевые слова PRIMARY KEY и AUTOINCREMENT.

Эти ключевые слова мы изучим чуть позже в нашем курсе. А пока вернёмся к команде CREATE TABLE и описаниям полей. Обязательных элементов в описании поля два:

1. Имя поля
2. Тип данных

Имя поля представляет собой текст, который может состоять из букв, цифр и некоторых специальных символов. Требования к именам полей различаются для разных типов баз данных, но можно с достаточной уверенностью сказать, что любая база данных примет в качестве имени поля комбинацию из латинских букв, цифр и символа подчёркивания, которая начинается не с цифры и не совпадает с каким-либо ключевым словом SQL. Именно такие названия полей используются в наших учебных базах данных.

С типами данных всё несколько сложнее. Каждый движок базы данных поддерживает свой набор типов данных, причём иногда типы данных, которые называются одинаково, реализуются в них по-разному.

Мы рассмотрим два набора типов данных:

- основные типы данных, поддержка которых рекомендована стандартом ANSI SQL,
- типы данных, поддерживаемые движком SQLite.

Типы данных ANSI SQL

ANSI SQL – это стандарт языка SQL, который рекомендован (но не обязателен) для использования в системах баз данных, поддерживающих работу с этим языком. В этом стандарте описан набор типов данных, на которые можно ориентироваться при изучении SQL. Мы рассмотрим эти типы, не очень сильно углубляясь в особенности их реализации, которые понятны только программистам.

Даже если вам придётся работать с системой баз данных, которая не полностью соответствует стандарту ANSI, знакомство со стандартными типами позволит вам быстро сориентироваться в том наборе типов данных, который поддерживается этой системой. Обычно производители таких систем включают в документацию таблицы соответствия, объясняющие, чем отличаются типы данных этой системы от стандартных.

Стандартные типы данных SQL можно условно разбить на несколько категорий:

- Числовые данные, предназначенные для хранения целых или дробных чисел.
- Символьные данные для хранения текстов разного размера.
- Временные данные для хранения даты, времени или временных интервалов.

- Логические значения, включающие всего два варианта: TRUE (Да) и FALSE (Нет)
- Произвольные данные. Эта категория включает несколько типов данных, которые часто приходится хранить в базах данных, но для работы с которыми обычно требуются дополнительные программы, не входящие в состав базы данных.

Каждая категория включает несколько типов данных, основные из которых мы сейчас перечислим.

Числовые данные

Эта категория включает разные типы данных для целых и дробных чисел. За время развития языка SQL их накопилось довольно много. Но прежде чем мы перейдём к рассмотрению типов числовых полей SQL, сделаем небольшое отступление: познакомимся с основными типами численных данных, с которыми могут работать современные вычислительные машины. Типы данных SQL однозначно им соответствуют.

Компьютеры позволяют хранить и обрабатывать численные данные трёх основных типов:

- целые числа;
- числа с плавающей точкой;
- числа с фиксированной точкой.

Для хранения **целых чисел** может выделяться разное количество памяти, от которого зависит минимальное и максимальное значение числа. Большинство компьютерных архитектур ориентировано на использование целых чисел размером 2, 4 и 8 байт. Если перевести это в привычные нам десятичные числа, то мы получим такие диапазоны:

- 2-байтовое целое число может принимать значения от -32 768 до 32 767
- 4-байтовое — от -2 147 483 648 до 2 147 483 647
- 8-байтовое — от -9 223 372 036 854 775 808 до 9 223 372 036 854 775 807

В зависимости от того, какие данные мы собираемся хранить, мы можем выбрать наиболее удобный для нас вариант. У «маленьких» целых чисел есть два основных преимущества. Во-первых, они занимают меньше места, что может быть важно при создании баз данных с очень большим количеством записей. Во-вторых, они обычно быстрее обрабатываются. Конечно, скорость обработки становится преимуществом, только если нам нужно выполнять большое количество вычислений.

Числа с плавающей точкой представляются в виде двух значений, которые называются мантиссой и порядком. Например, числа 0.0031415 и 3141500000000 можно представить в таком виде:

- $3.1415 * 10^{-3}$
- $3.1415 * 10^{8}$

У этих двух чисел одинаковые мантиссы (3.1415), но разные порядки (-3 и 8). Для хранения этих двух чисел нужен одинаковый объём памяти, но, очевидно, точность их представления зависит от порядка. Во многих случаях это оправданно. Например, вы можете хранить в базе данных описания космических объектов с указанием их массы, которая может лежать в диапазоне от нескольких грамм до миллиардов тонн, и в каждом случае полученной точности будет достаточно для практических нужд.

Но такое представление вряд ли подойдёт для денежных сумм. И многомиллиардные, и копеечные суммы нужно хранить с определённой, заранее известной точностью: «до копейки». Когда точность представления так важна, используют третий вид чисел: **числа с фиксированной точкой**.

Для таких чисел всегда указывается точное количество знаков после запятой. Например, денежные суммы можно представлять в виде дробного числа в виде рублей с копейками (или долларов с

центами, или тенге с тиынами — и т. п., в зависимости от валюты). При этом у вас будет всегда два знака после запятой, причём суммы нельзя округлять, так как каждая копейка должна быть учтена.

После этого краткого эскурса в архитектуру компьютеров мы можем перейти к типам данных ANSI SQL.

Начнём с целых чисел. Для них используется несколько типов, определяющих размер выделяемой памяти, а значит, и допустимый диапазон значений числа.

- **INTEGER** (допускается сокращение **INT**) – основной тип данных для хранения целых чисел, в том числе отрицательных. В большинстве современных реализаций баз данных для этого типа используются 4-байтные или 8-байтные числа. В наших учебных базах данных мы использовали этот тип данных, потому что его диапазона значений нам заведомо хватает для наших целей.
- **BIGINT** – тип данных для хранения больших целых чисел (на современных системах чаще всего используется 8-байтные целые числа).
- **SMALLINT** – тип данных для хранения сравнительно небольших целых чисел. Как правило, для представления данных этого типа используются 2-х или 4-байтные числа.

Как вы поняли, размер выделяемой памяти и, следовательно, диапазон значений для этих типов зависит от используемой системы базы данных. Если вам нужно выбрать тип данных, соответствующий вашим требованиям к диапазону, нужно свериться с документацией по той системе баз данных, которую вы используете. С уверенностью можно сказать только, что во всех системах выполняется такое соотношение:

размер **SMALLINT** <= размер **INTEGER** <= размер **BIGINT**.

Для хранения чисел с фиксированной точкой в ANSI SQL определены следующие типы данных.

- **DECIMAL** (допустимо сокращение **DEC**)
- **NUMERIC**

Между этими типами данных в стандарте ANSI SQL есть тонкие различия, на которых мы не будем останавливаться, тем более что большинство существующих систем баз данных их игнорирует. При описании поля одного из этих типов после ключевого слова в скобках дополнительно указываются максимальное общее количество цифр числа, а также количество цифр после запятой. Например, следующее описание поля определяет, что в нём будут храниться числа, содержащие до 12 цифр, из которых две всегда находятся после запятой.

```
Money DECIMAL(12, 2)
```

В поле такого типа вы можете хранить суммы в диапазоне от 1 копейки (0.01) до 10 миллиардов рублей без одной копейки (9 999 999 999.99), и точность представления этих сумм будет одинаковой, буквально «до копейки».

Для хранения данных с плавающей точкой, как и в случае с целыми числами, ANSI SQL поддерживает несколько типов данных, которые различаются размером выделяемой под них памяти. Конкретный тип выбирается в зависимости от требуемой точности и диапазона представления, причём в разных реализациях минимальные и максимальные значения могут быть разными. Мы просто перечислим эти типы данных в порядке возрастания точности:

- **REAL** – тип данных для чисел с плавающей точкой сравнительно невысокой точности;
- **FLOAT** – тип данных с плавающей точкой для чисел с настраиваемой точностью (в некоторых пределах);
- **DOUBLE PRECISION** – тип данных для чисел с плавающей точкой сравнительно высокой точности.

Настраиваемая точность для типа FLOAT означает, что необходимую точность можно указать в описании поля. Способ указания точности очень похож на тот, который используется для DECIMAL. Но отличие в том, что для типа FLOAT точность указывается не в количестве десятичных цифр, а в битах:

```
Weight FLOAT(52,4)
```

Чтобы понять, что означают эти цифры, нужно углубляться в способы представления чисел с плавающей точкой в памяти компьютера. А это уже территория программистов, на которую мы не будем заходить.

Символьные данные

Стандарт ANSI SQL определяет такие основные типы данных для хранения текстовой информации.

- **CHARACTER** (допустимо сокращение **CHAR**) – строка фиксированной длины. Длина строки указывается в скобках после ключевого слова CHAR, например:

```
Category CHAR(30)
```

В этом примере описано поле Category, которое может содержать текст длиной до 30 символов. Текст длиной меньше 30 символов дополняется справа пробелами.

- **CHARACTER VARYING** (сокращённо **VARCHAR**) – строка переменной длины. Максимальная длина также указывается в скобках:

```
Description VARCHAR(100)
```

В отличие от типа CHAR, строки типа VARCHAR не дополняются справа пробелами, а содержат ровно столько символов, сколько требуется для сохранения строки текста. Например, в следующем запросе INSERT используется строка 'Рестораны' размером 9 символов.

```
insert into Categories (Category) values ('Рестораны')
```

Если поле Category имеет тип VARCHAR (30), то будут сохранены 9 символов, а если его тип CHAR(30), то к этим 9 символам справа будет добавлен ещё 21 пробел.

Возможно, вы задали себе вопрос: зачем нужно два типа для одних и тех же, по сути данных? Какой тип лучше выбрать для хранения строки до 100 символов — CHAR(100) или VARCHAR(100)?

В большинстве систем БД тип CHAR занимает больше места, чем VARCHAR, но при этом запросы с данными типа CHAR быстрее обрабатываются. Но это различие тоже становится важным только в базах данных, хранящих большие объёмы информации.

С типом CHAR (и VARCHAR) есть одна проблема: в разных языках для хранения отдельных символов может использоваться разный объём памяти. В большинстве реализаций баз данных размер CHAR соответствует количеству байт, которое отводится под значение поля. Этого достаточно для символов латинского алфавита, но совсем недостаточно, например, для китайского или японского языков, в которых отдельный символ-иероглиф может представляться в виде нескольких байт. Поэтому стандарт ANSI SQL предусматривает ещё пару типов данных для хранения текстовых строк:

- **NATIONAL CHARACTER** (сокращённо **NCHAR**) — строка национальных символов фиксированной длины
- **NATIONAL VARYING CHARACTER** (сокращённо **NCHAR VARYING**) — строка национальных символов переменной длины

При использовании этих типов можно быть уверенным, что для поля выделяется память, достаточная для хранения указанного количества символов того языка, который указан в настройках базы данных.

Временные данные

Основными типами данных для представления информации о времени в стандарте ANSI SQL являются:

- **DATE** – тип данных для хранения дат. В запросах SQL для представления конкретной даты используется текстовая строка вроде '2015-01-28', но форматы этих строк обычно зависят от множества настроек. Эту же дату можно представить в виде '28/01/2014' или 'Jan 28 2015' или '28 января 2015 г.', из-за чего у пользователей баз данных

нередко возникают затруднения с составлением запросов. Из-за этой путаницы программисты иногда предпочитают вместо типа DATE использовать символьный тип CHAR, самостоятельно определяя неизменяемый формат представления даты (например, '20150128').

- **TIME** – тип данных для хранения времени. С форматом представлением времени дела обстоят несколько проще. Время можно представить, например, такой строкой: '18:30:05'. В этом примере указано время «18 часов 30 минут 5 секунд». Секунды можно опустить, например: '18:30'.
- **TIMESTAMP** – тип данных для хранения временной отметки, которая включает в себя и дату, и время. Эта отметка представляется в виде целого числа, отсчитывающего количество единиц времени, прошедших от фиксированной даты. Используемые единицы и фиксированная дата зависят от системы базы данных. Например, может использоваться так называемое «время UNIX»: количество секунд, прошедших с полуночи 1 января 1970 года. Этот тип данных очень часто используют в базах данных для регистрации точного времени наступления различных событий. Например, если вы используете интернет-клиент для работы со своим счётом в банке, информация о каждой вашей операции может сохраняться в базе данных вместе с датой и временем её выполнения в формате TIMESTAMP.

Логические данные

Для представления логических данных стандарт ANSI SQL описывает всего один тип:

- **BOOLEAN** – тип данных, которые могут принимать всего два значения: **TRUE** («истина» или «да») и **FALSE** («ложь» или «нет»). Данные такого вида так часто используются в компьютерных программах, что для них определили собственный тип. В нашей учебной базе данных мы могли бы использовать этот тип данных для поля **Done** таблицы **Tasks**.

Произвольные данные

По мере развития и всё более широкого применения баз данных стала возникать необходимость хранить в них всё более и более разнообразные данные: документы в различных форматах, компьютерные изображения, записи звука и видео и т. п. Отличительной особенностью этих данных является то, что для работы с ними обычно используются специальные программы. Для базы данных это просто длинный набор байт, который она должна сохранить и вернуть по запросу в неизменном виде. Для таких данных в ANSI SQL был добавлен специальный тип:

- **BINARY LARGE OBJECT** (сокращённо **BLOB**) – массив двоичных данных произвольной структуры. Мы не будем работать с такими данными при изучении этого курса. Информация об этом типе приведена в нашем курсе только для того, чтобы вы знали, что в современных базах данных можно хранить всё, что угодно.

Типы данных SQLite

Если перечень типов данных ANSI SQL вызвал у вас лёгкую панику, то вы не одиноки (и это мы ещё перечислили не все типы, рекомендованные стандартом). Изучение разнообразных типов данных и способов их использования — это одна из самых сложных и запутанных тем при знакомстве с любой системой баз данных.

Ситуация на самом деле ещё хуже: в дополнение (или взамен) стандартных типов данных каждая система предлагает множество своих собственных типов, различия между которыми часто не очевидны. Предполагается, что это многообразие типов позволяет делать «тонкую настройку» баз данных, выбирая для каждого случая идеальный тип данных, обеспечивающий максимальную скорость работы базы данных или экономию объёма памяти — в зависимости от того, что важнее.

Но нам совершенно ни к чему знакомиться с этими тонкими настройками в нашем курсе. Именно поэтому для учебных баз данных был выбран движок SQLite, разработчики которого радикально решили проблему многообразия типов, не нужного в большинстве простых программ, предназначенных для одного пользователя.

В SQLite используются три основных принципа, упрощающих работу с типами данных.

Во-первых, всё разнообразие типов сводится к четырём так называемым классам хранения:

- **INTEGER** – для целых чисел размером до 8 байт.
- **REAL** – для чисел с плавающей точкой (для преставления всегда используется 8 байт, что соответствует стандартному типу **DOUBLE PRECISION**).
- **TEXT** – для текстовых данных.
- **BLOB** – для хранения произвольных данных.

На самом деле, есть ещё пятый класс хранения **NULL**, с назначением которого мы познакомимся позже в этом курсе.

Во-вторых, для хранения данных выделяется столько памяти, сколько необходимо для представления конкретного значения. Нет больше необходимости разбираться в различиях между INT, BIGINT и SMALLINT: для значения поля в каждой записи будет выделено столько байт, сколько нужно для представления числа с максимальной точностью. Это же касается текстовых данных: значение поле типа TEXT может содержать от нуля до миллиона символов, а если миллиона недостаточно, то максимальный размер может быть переопределен настройками базы данных.

В-третьих, тип данных в SQLite является не обязательным, а декларативным. Это означает, что в любом поле могут храниться данные любого типа. Когда мы сохраняем данные в базу, движок SQLite пытается преобразовать их в тот класс хранения, который указан для поля. Но если преобразование не может быть выполнено, то данные сохраняются в неизменном виде.

При создании таблиц с помощью команды CREATE TABLE в базе данных SQLite могут указываться любые типы данных, включая как стандартные типы ANSI SQL, так и нестандартные типы самых популярных систем баз данных. Движок SQLite по указанному типу определяет класс хранения с помощью набора несложных правил. В следующей таблице представлено соответствие стандартных типов ANSI SQL классам хранения SQLite, которые для них используются.

Тип данных ANSI SQL	Класс хранения SQLite
INT INTEGER SMALLINT BIGINT BOOLEAN TIMESTAMP	INTEGER
CHARACTER VARCHAR NCHAR NVARCHAR DATE TIME	TEXT
BLOB	BLOB
REAL DOUBLE	REAL

DOUBLE PRECISION	
NUMERIC DECIMAL	INTEGER или REAL (в зависимости от значения)

В SQLite нет встроенных типов данных для представления даты и времени. Хотя эти типы и приведены в таблице, для хранения таких данных в разных ситуациях может использоваться как текстовое, так и численное представление. Некоторые особенности работы с датой и временем мы узнаем позже в этом курсе, когда будем изучать встроенные функции SQL.

Классы хранения SQLite можно использовать для обозначения типа поля при создании таблиц. В наших учебных базах данных использовались именно они, как в приведенной уже команде создания таблицы Categories:

```
create table Categories
(Category_ID INTEGER,
Category TEXT,
Category_Description TEXT)
```

Типы данных SQL — тест

Какой тип данных лучше использовать для хранения данных о денежных суммах на банковских счетах?

Выберите один ответ:

- a. BOOLEAN
- b. SMALLINT
- c. NUMERIC
- d. REAL

Какой тип данных предполагает дополнение строки пробелами до 50 символов?

Выберите один или несколько ответов:

- a. NCHAR VARYING(50)
- b. VARCHAR(50)
- c. NCHAR(50)
- d. CHAR(50)

Какой тип данных предполагает хранение в одном поле и даты, и времени?

Выберите один или несколько ответов:

- a. TIMESTAMP
- b. TIME
- c. DATE