Как устроены таблицы баз данных

Таблицы

Вся информация в базах данных хранится в виде таблиц.

Например, можно изобразить таблицу базы данных в таком виде.

Таблица **Подружки**

Имя	**Телефон**	**День рождения**	**Примечание**
Лена	9101234567	5 мая	Любит розы
Катя	9037654321	10 октября	Ненавидит розы
Наташа	9817777777		Подруга Кати

У каждой таблицы в базе данных есть своё название или имя. В одной базе данных не может быть двух таблиц с одинаковыми именами. Наша таблица называется **Подружки**.

Записи

Таблицы состоят из записей. Их ещё называют строками. Это название не совсем правильно, но зато наглядно: мы смотрим на представленную выше таблицу и сразу понимаем, о чём идёт речь.

Все записи одной таблицы имеют одинаковую структуру. Мы можем оставить пустой ячейку «день рождения» в записи, если не знаем правильной даты рождения. Но мы не можем использовать её для хранения какой-то другой информации. Мы также не можем создать запись, в которой этойячейки нет вообще.

Здесь надо сделать оговорку: существуют системы баз данных, которые позволяют такие вольности со структурой записей. Но язык SQL не предназначен для работы с такими базами данных, поэтому больше мы о них в этом курсе вспоминать не будем.

Поля таблиц

Записи таблицы состоят из отдельных полей. Этот набор полей называют структурой таблицы. В нашем примере структура таблицы включает четыре поля: **Имя**, **Телефон**, **День рождения** и **Примечание**.

В принципе, знания имени таблицы и названий полей уже достаточно для того, чтобы начать использовать язык SQL для работы с данными. Например, мы можем использовать такой запрос, чтобы найти номер телефона по имени подружки:

```
select Телефон from Подружки where Имя = 'Лена'
```

Или наоборот, мы можем найти имя по номеру телефона:

```
select Имя from Подружки where Телефон = 9037654321
```

Но, прежде чем мы окунёмся в мир запросов SQL, выясним ещё кое-какие детали.

Типы полей

Поля таблиц могут быть предназначены для хранения данных разных типов. Тип каждого поля задаётся при создании таблицы. В нашем простейшем примере мы могли бы использовать такие типы полей:

- Текст — для полей **Имя** и **Примечание**
- Число — для поля **Телефон**
- Дата — для поля **День рождения**

Тип поля накладывает ограничения на данные, которые мы можем в нём хранить. Например, если мы выбрали тип Число для поля Телефон, то мы уже не можем хранить в нём произвольный текст: данные этого типа могут состоять только из цифр.

Это же относится к типу Дата: поле **День рождения** с таким типом будет предназначено только для хранения дат, и больше ни для чего. При попытке создать запись со значением, которое не соответствует типу поля, база данных вернёт нам сообщение об ошибке. Здесь, конечно, есть множество нюансов, но о них мы узнаем позже в этом курсе.

Связи между таблицами

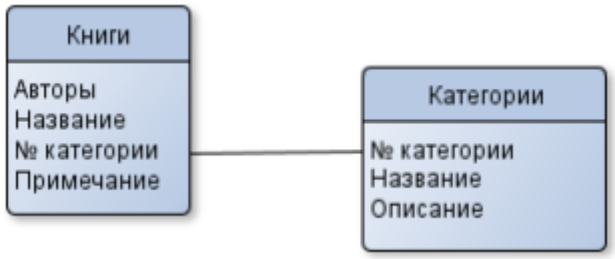
Обычно базы данных состоят из множества таблиц, записи которых могут быть связаны между собой. Чтобы понять, как они связываются, нам нужно перечитать самую первую фразу этой главы:

```
Вся информация в базах данных хранится в виде таблиц.
```

Связи между записями разных таблиц — это тоже информация. А значит, связи тоже хранятся в таблицах.

Для хранения связей между записями таблиц обычно используют дополнительные поля. Давайте рассмотрим ещё один пример, чтобы понять, как это делается.

Следующая иллюстрация описывает структуру простой базы данных для учёта книг домашней библиотеки, состоящей из двух таблиц: **Книги** и **Категории**.



Библиотека разделена на категории, и каждая книга входит в одну из этих категорий. Категории могут быть, например, такими: Фантастика, Романы, Бизнес-книги, Программирование.

Данные в таблицах могут быть, например, такими.

Таблица **Категории**

№ категории	Название	Описание
1	Фантастика	НФ, Фэнтэзи
2	Романы	Женские и приключенческие романы
3	Бизнес-книги	Про бизнес и тайм-менеджмент
4	Программирование	Языки программирования, языки разметки, базы данных

Таблица **Книги**

Авторы	Название	№ категории	Примечание
Моран Брайан	12 недель в году	3	Перечитать
Страуструп Бьёрн	Язык программирования C++	4	Дал почитать Олегу
Фридл Джеффри	Регулярные выражения	4	

Как видим, в обеих таблицах присутствует поле **№ категории**. Именно это поле связывает записи таблицы **Книги** с записями таблицы **Категории**: книга «12 недель в году» относится к категории 3 (Бизнес-книги), а книги «Язык программирования C++» и «Регулярные выражения» – к категории 4 (Программирование).

Очевидно, что в таблице Категории значения этих полей должны быть уникальными (то есть не может быть двух записей с одинаковым значением этого поля). В этом курсе мы научимся сообщать базе данных, что она должна сама следить за уникальностью значений таких полей.

Имена полей, по которым определяется связь, не обязательно должны быть одинаковыми в разных таблицах, но это удобный способ дополнительно показать связь между полями человеку, который изучает структуру базы данных.

В нашем примере каждая книга может относиться только к одной категории. Разработчики баз данных выразили бы это так: между таблицами **Категории** и **Книги** существует связь вида «один-ко-многим» (одна категория ко многим книгам). О том, какие ещё бывают связи, и как они реализуются в таблицах, мы тоже узнаем из этого курса.

Азы SQL

В этом разделе вы познакомитесь с базовыми конструкциями языка SQL, предназначенными для манипулирования данными.

Раздел познакомит вас с операторами SELECT, INSERT, UPDATE и DELETE в их наиболее часто используемых формах, не зависящих от типа базы данных. Этих знаний достаточно для того, чтобы понять, как люди и программы взаимодействуют с базами данных для получения и изменения хранящейся в них информации.

Учебные базы данных

В нашем курсе мы будем использовать две сравнительно простые базы данных, состоящие из нескольких таблиц. Несмотря на простоту, эти базы данных вполне могли бы использоваться в реальных полезных программах.

Почему именно две? Потому что мы будем использовать следующий подход: в уроках этого курса все изучаемые запросы SQL будут иллюстрироваться на одной из этих баз данных, а все упражнения для закрепления пройденного вы будете выполнять с другой базой. Так вы гарантированно получите навык самостоятельного составления запросов.

Давайте познакомимся со структурой этих баз данных.

Но, прежде чем мы перейдём к рассмотрению их структуры, сделаем одну оговорку. Мы будем использовать английские слова для названий таблиц и их полей. Многие современные базы данных позволяют использовать для этих целей практически любые языки, включая русский, но пока ещё далеко не все.

Учебная база данных №1. Списки дел

Наша первая база данных предназначена для хранения повседневных задач, упорядоченных в виде списков. Она может быть использована для планирования личных дел.

В таблицах базы данных будут храниться следующие объекты (или сущности).

Задача (task) — описание задачи, которую нужно выполнить. Например: «записаться в бассейн» или «написать курсовую».

Список задач (tasklist) — используется для группировки задач, относящихся к одной теме или одному проекту. Например, можно создать список «кандидатская диссертация» и включать в него все задачи, относящиеся к написанию и защите диссертации.

Категория (category) — довольно широкие сферы жизни, к которым можно отнести решаемые нами задачи. Примеры категорий: семья, здоровье, свой бизнес.

Мы будем использовать следующие правила:

1. Каждая задача может входить только в один список задач.
2. Каждый список задач относится к одной категории.

Эти правила важно оговорить заранее, потому что они определяют структуру нашей базы данных. Если эти правила кажутся вам странными и оторванными от реальной жизни, это не страшно: после изучения этого курса вы будете знать, как можно спроектировать структуру базы данных с использованием любых других правил.

Итак, наша учебная база данных состоит из таких таблиц.

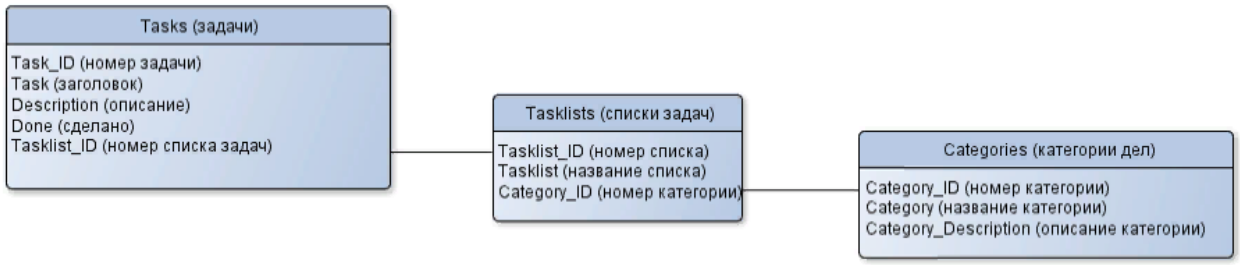


Таблица **Tasks** (задачи) включает такие поля

Поле	**Описание**	**Тип поля**
Task_ID	Порядковый номер задачи (каждой новой задаче автоматически присваивается уникальный номер)	Целое число
Task	Краткое описание (заголовок) задачи.	Текст
Description	Подробное описание задачи. Может быть пустым (для описания большинства простых задач достаточно заголовка).	Текст
Done	Признак «Сделано» – Нет, если задача ещё не выполнена, Да, если выполнена.	Целое число, представляющее логическое значение (1 - да / 0 - нет)
Tasklist_ID	Порядковый номер списка, в который входит задача (см. следующую таблицу)	Целое число

Таблица **Tasklists** (списки задач) включает поля

Поле	**Описание**	**Тип поля**
Tasklist_ID	Порядковый номер списка (каждому новому списку автоматически присваивается уникальный номер)	Целое число
Tasklist	Название списка задач	Текст
Category_ID	Категория, к которой относится этот список задач	Целое число (см. следующую таблицу)

Таблица **Categories** (категории) включает поля

Поле	**Описание**	**Тип поля**
Category_ID	Порядковый номер категории (каждой новой категории автоматически присваивается уникальный номер)	Целое число
Category	Название категории	Текст

Поле	Описание	Тип поля
Category_Description	Описание категории	Текст

Вы видите, что в каждой таблице есть поле, содержащее номер записи: Task_ID, Tasklist_ID, Category_ID. **ID** в данном случае — это сокращения от слова Identifier (идентификатор). Это сокращение очень часто используется в именах полей баз данных, предназначенных для однозначной ссылки на конкретную запись (или, другими словами, для *идентификации* записей).

Учебная база данных №2. Учёт расходов

Вторая учебная база данных предназначена для учёта домашних расходов. С помощью этой базы можно понять, на что тратятся деньги из домашнего бюджета.

В базе будет сохраняться информация обо всех совершаемых покупках. Покупки классифицируются по категориям (например: еда, одежда, развлечения и т. п.)

При этом дополнительно можно вести специальный учёт по отдельным видам товаров (чтобы узнать, например, сколько тратится денег на конфеты, на книги или на компьютерные игры).

База данных состоит из четырёх таблиц.

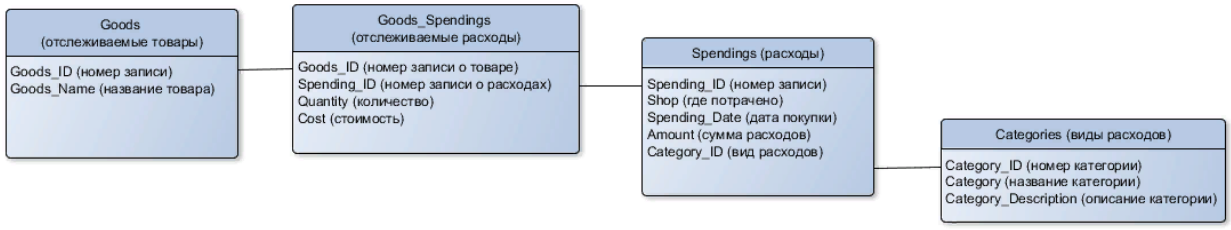


Таблица **Spendings** (расходы) предназначена для учёта всех совершаемых расходов и включает такие поля.

Поле	Описание	Тип поля
Spending_ID	Порядковый номер покупки (каждой новой записи автоматически присваивается уникальный номер)	Целое число
Shop	Название магазина, в котором сделана покупка. «Магазин» в данном случае — условное название: это может быть и ресторан, и банк, и компьютерная игра, словом, то учреждение, в пользу которого вы расстаётесь со своими деньгами.	Текст
Spending_Date	Дата покупки	Дата
Amount	Сумма всей покупки. Например, вся сумма, потраченная в ресторане, или сумма чека из продуктового магазина. Если по какому-то товару ведётся дополнительный учёт, его стоимость всё равно включается в сумму покупки. Учёт ведётся в рублях с копейками.	Число

Поле	Описание	Тип поля
Category_ID	Категория, к которой относится эта покупка	Целое число (ссылается на запись в таблице Categories)

Структура таблицы **Categories** ничем не отличается от одноимённой таблицы первой базы данных.

Поле	Описание	Тип поля
Category_ID	Порядковый номер категории (каждой новой категории автоматически присваивается уникальный номер)	Целое число
Category	Название категории	Текст
Category_Description	Описание категории	Текст

Таблица **Goods** (товары) содержит список товаров, по которым ведётся дополнительный детальный учёт, и имеет очень простую структуру.

Поле	Описание	Тип поля
Goods_ID	Порядковый номер товара (каждой новой записи автоматически присваивается уникальный номер)	Целое число
Goods_Name	Название товара	Текст

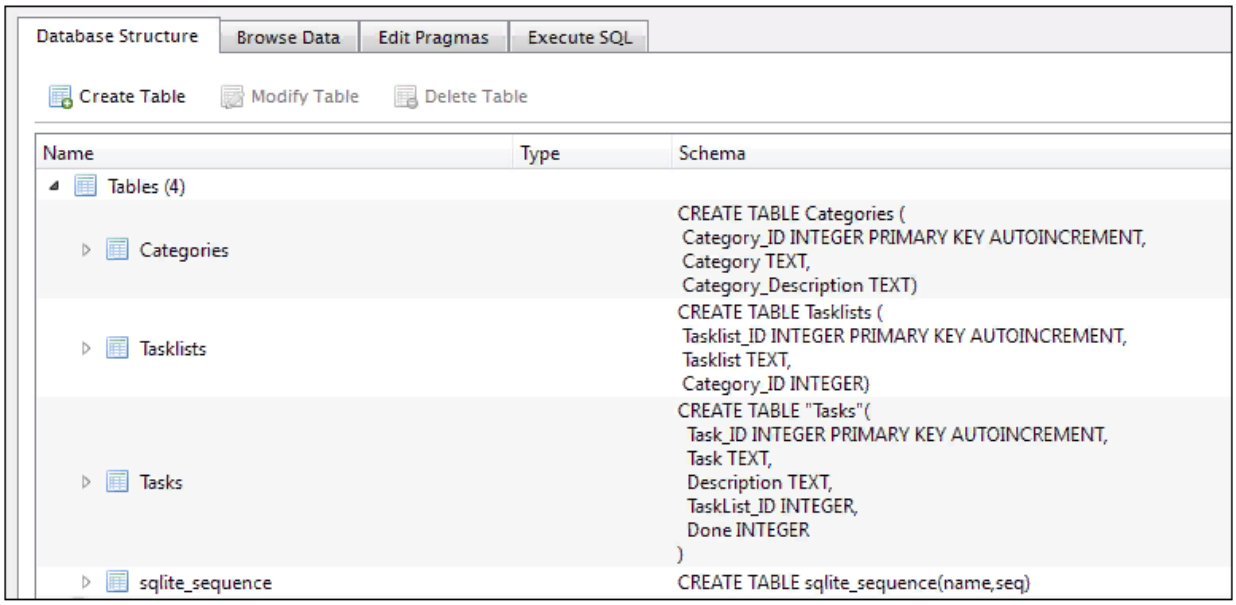
Наконец, таблица **Goods_Spendings** предназначена для ведения дополнительного учёта по товарам из списка, представленного в таблице **Goods**. В таблице **Goods_Spendings** сохраняется информация о количестве купленного товара, его стоимости, а также о покупке, в рамках которой был приобретён этот товар.

Поле	Описание	Тип поля
Goods_ID	Порядковый номер товара в таблице **Goods**	Целое число
Spending_ID	Порядковый номер покупки в таблице **Spendings**, в рамках которой куплен этот товар	Целое число
Quantity	Количество купленного товара в тех единицах, которые удобны для учёта. Например, шоколадки удобно учитывать в штуках, вино – в бутылках, разливное пиво – в литрах.	Целое число
Cost	Стоимость всего товара в пределах одной покупки. То есть это не цена за штуку / бутылку / литр, а общая сумма, потраченная на товар в рамках покупки.	Число с двумя знаками после запятой

файлы описанных здесь баз tasks-start.sqlite, spendings-start.sqlite

В этих файлах уже содержится по несколько записей в каждой таблице, чтобы мы могли сразу использовать их для выполнения запросов, которые мы начнём изучать в следующей главе курса.

Установите программу SQLite Database Browser, если вы этого до сих пор не сделали. Откройте с её помощью первую базу данных, посмотрите на структуру таблиц (на вкладке Database Structure) и на их содержимое (на вкладке Browse Data).



У вас наверное, возник вопрос: почему таблиц четыре, а не три?

Таблица **sqlite_sequence** – это служебная таблица, которую движок SQLite создал самостоятельно. Она используется для сохранения последних присвоенных значений номерам записей, которые в нашей базе данных хранятся в полях Task_ID, Tasklist_ID и Category_ID. Например, каждый раз, когда мы добавляем новую запись в таблицу Tasks, движок обращается к этой таблице, берёт сохранённое значение, увеличивает его на единицу, и присваивает полученный номер новой записи в поле Task_ID.

Программы, предназначенные для управления базами данных часто создают такие служебные таблицы для собственных нужд.

Запросы SELECT

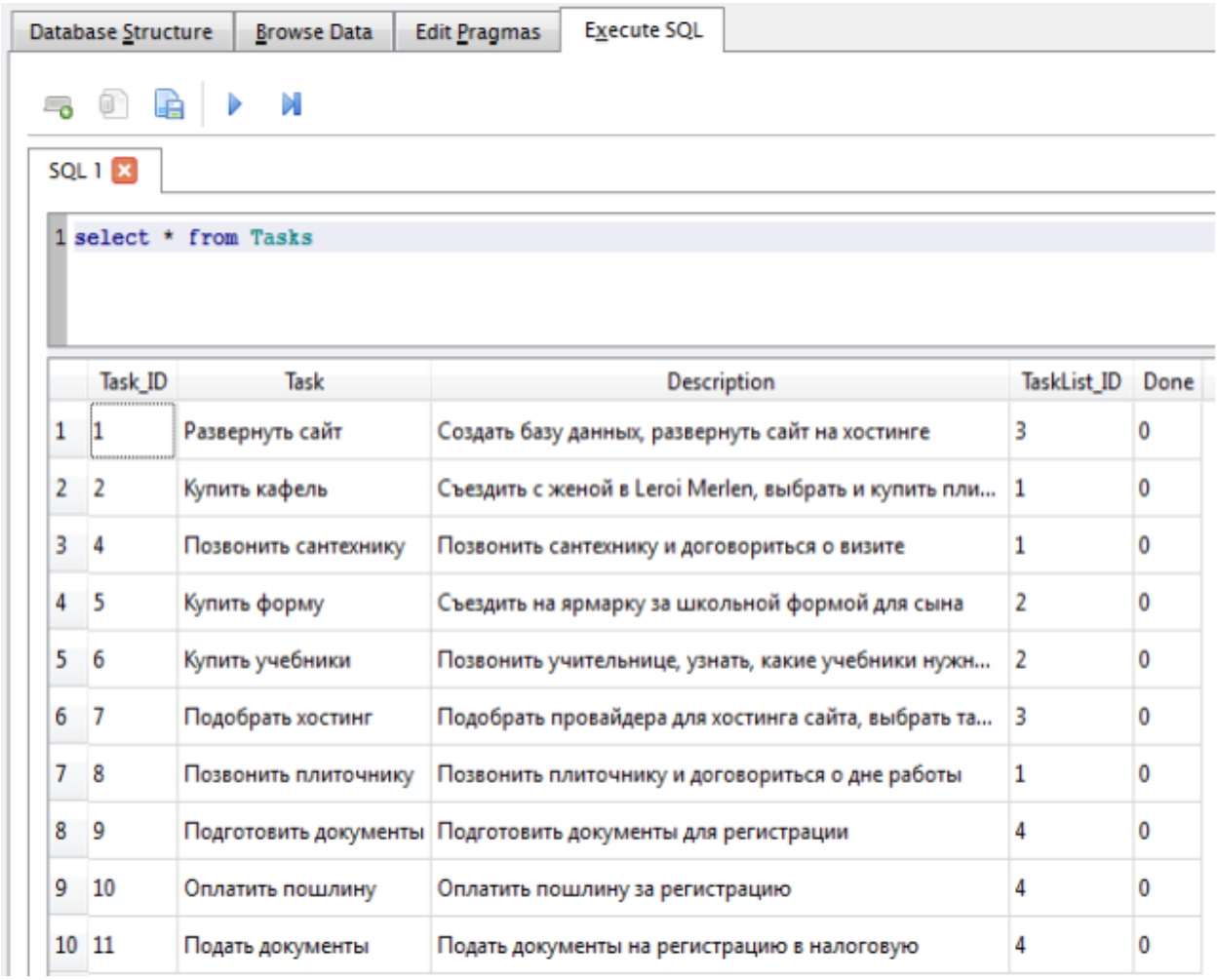
Итак, у нас есть база данных и программа для работы с ней. Давайте же, не мешкая, выполним наш первый запрос на языке SQL.

Запустите программу DB Browser for SQlite и откройте учебную базу данных tasks-start.sqlite. Перейдите на вкладку Execute SQL, введите в текстовом поле такой запрос

```
select * from Tasks
```

и нажмите кнопку  или клавишу F5.

Программа выведет на экран результат выполнения запроса.



	Task_ID	Task	Description	TaskList_ID	Done
1	1	Развернуть сайт	Создать базу данных, развернуть сайт на хостинге	3	0
2	2	Купить кафель	Съездить с женой в Leroi Merlen, выбрать и купить пли...	1	0
3	4	Позвонить сантехнику	Позвонить сантехнику и договориться о визите	1	0
4	5	Купить форму	Съездить на ярмарку за школьной формой для сына	2	0
5	6	Купить учебники	Позвонить учительнице, узнать, какие учебники нужн...	2	0
6	7	Подобрать хостинг	Подобрать провайдера для хостинга сайта, выбрать та...	3	0
7	8	Позвонить плиточнику	Позвонить плиточнику и договориться о дне работы	1	0
8	9	Подготовить документы	Подготовить документы для регистрации	4	0
9	10	Оплатить пошлину	Оплатить пошлину за регистрацию	4	0
10	11	Подать документы	Подать документы на регистрацию в налоговую	4	0

Поздравляю! Вы выполнили свой первый запрос SELECT – самый популярный вид запроса, ради которого и были придуманы базы данных. Запросы SELECT предназначены для получения данных, которые хранятся в базе.

Давайте разберёмся, что же мы запросили.

Наш запрос можно представить в таком обобщённом виде:

```
SELECT {список полей} FROM {имя таблицы}
```

Фигурные скобки здесь показывают, что их содержимое (вместе со скобками) должно быть заменено на конкретные элементы базы данных — в нашем случае это список полей таблицы и её имя. Мы на протяжении всего курса будем использовать такую форму записи.

Слова **SELECT** (выбрать) и **FROM** (из) — это элементы языка SQL, так называемые *ключевые слова*.

В нашем запросе вместо списка полей стоит символ звёздочки. Это тоже элемент языка SQL. Звёздочка в этой позиции означает, что мы хотим получить данные из всех полей таблицы. В этом случае нам даже не нужно знать, из каких полей состоит таблица.

Язык SQL разрабатывался таким образом, чтобы быть похожим на естественный английский язык, поэтому наш запрос похож на фразу английского языка «выбрать всё из таблицы Tasks».

Язык SQL нечувствителен к регистру букв. Мы могли бы написать слово **SELECT** так: **select**, **Select** или даже **SeLeCt**. В дальнейшем при описании правил составления запросов мы будем использовать верхний регистр (заглавные буквы), чтобы визуально выделять ключевые слова, а в примерах запросов — нижний регистр (строчные буквы), потому что так их легче читать и вводить.

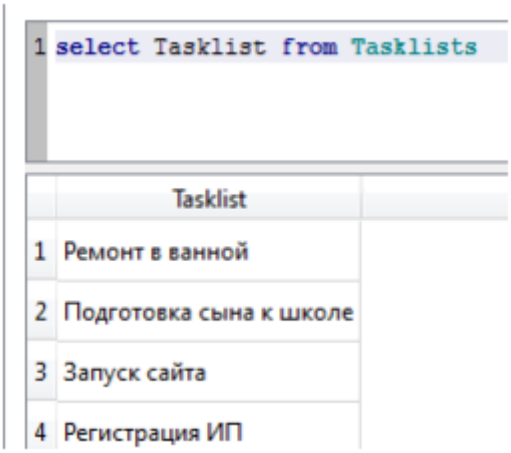
Вы уже поняли, как можно получить данные из других таблиц. Введите и выполните такие запросы, чтобы получить содержимое таблиц Tasklists и Categories:

```
select * from Tasklists
select * from Categories
```

Выбор нужных полей

Если нам нужно получить значения не всех, а только одного поля, то в запрос нужно подставить его имя вместо звёздочки. Следующий запрос вернёт нам перечень всех списков задач.

```
select Tasklist from Tasklists
```

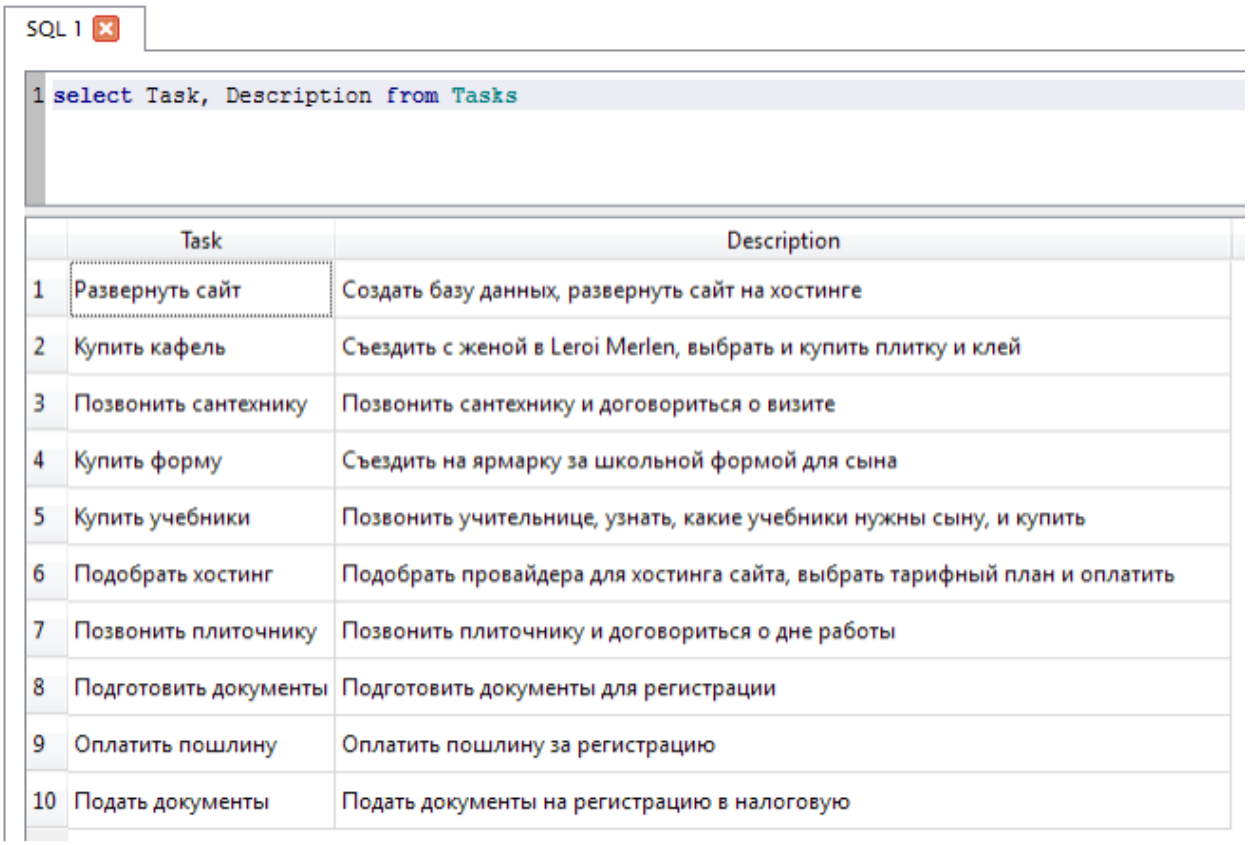


```
1 select Tasklist from Tasklists
```

	Tasklist
1	Ремонт в ванной
2	Подготовка сына к школе
3	Запуск сайта
4	Регистрация ИП

Если нужно получить значения нескольких полей, то они разделяются запятой. Следующий запрос вернёт нам заголовки всех задач с их описанием.

```
select Task, Description from Tasks
```



SQL 1

```
1 select Task, Description from Tasks
```

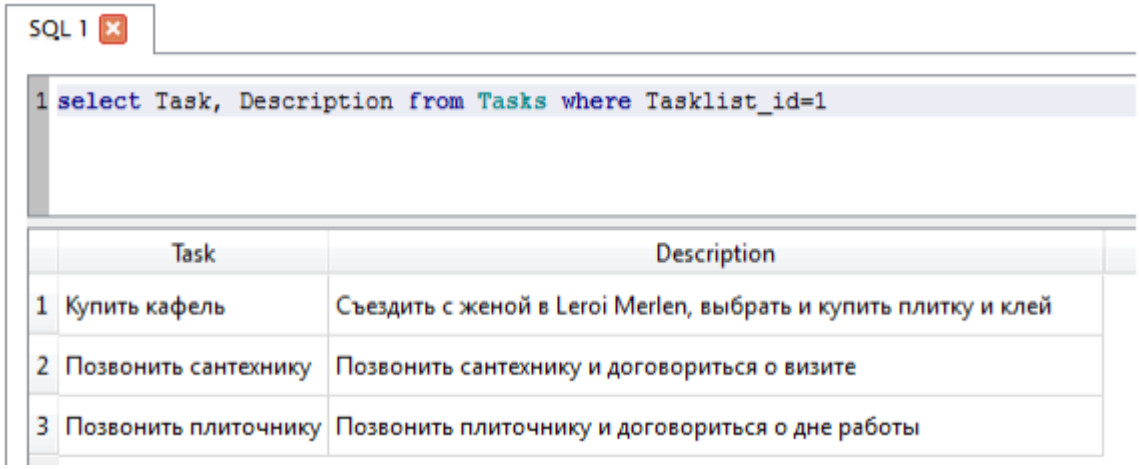
	Task	Description
1	Развернуть сайт	Создать базу данных, развернуть сайт на хостинге
2	Купить кафель	Съездить с женой в Leroi Merlen, выбрать и купить плитку и клей
3	Позвонить сантехнику	Позвонить сантехнику и договориться о визите
4	Купить форму	Съездить на ярмарку за школьной формой для сына
5	Купить учебники	Позвонить учительнице, узнать, какие учебники нужны сыну, и купить
6	Подобрать хостинг	Подобрать провайдера для хостинга сайта, выбрать тарифный план и оплатить
7	Позвонить плиточнику	Позвонить плиточнику и договориться о дне работы
8	Подготовить документы	Подготовить документы для регистрации
9	Оплатить пошлину	Оплатить пошлину за регистрацию
10	Подать документы	Подать документы на регистрацию в налоговую

Выбор нужных записей

Запросы, с которыми мы познакомились, просты, но не очень полезны. Довольно редко нам приходится просматривать все записи таблицы подряд. Гораздо чаще нужно отобрать только часть записей по какому-то признаку. Например, если мы работаем со списками задач, то хотелось бы просмотреть только те задачи, которые относятся к нужному списку.

Конечно, язык SQL предоставляет нам такую возможность. Следующий запрос вернёт нам заголовки и описания задач, которые относятся только к списку с номером 1.

```
select Task, Description from Tasks where Tasklist_ID=1
```



Мы видим, что в этом запросе появилась новая конструкция, начинающаяся с ключевого слова **WHERE** (где).

Обобщённо запрос можно представить так:

```
SELECT {список полей} FROM {имя таблицы} WHERE {условие отбора}
```

В качестве условия отбора записей использовано *выражение*:

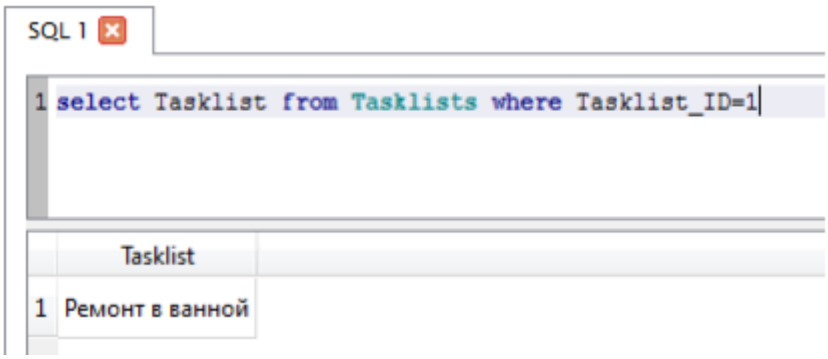
```
Tasklist_ID=1
```

Это условие, как вы, наверное, уже догадались, указывает на то, что нужно отобрать только те записи, в которых значение поля Tasklist_ID имеет значение 1.

Но что это за список задач с номером 1? Не должны же мы запоминать номера всех списков задач!

Конечно, не должны — для этого у нас есть база данных. Мы можем обратиться к ней с запросом, чтобы она подсказала нам, о каком списке идёт речь:

```
select Tasklist from Tasklists where Tasklist_ID=1
```



В дальнейшем мы узнаем, как избежать запоминания и ввода номеров записей, а использовать вместо этого более привычные и легко запоминаемые текстовые названия. А пока выполните упражнения со второй учебной базой данных, чтобы закрепить пройденное.

Практика

Для выполнения задания скачайте учебную базу данных «Учёт расходов» tasks-start.sqlite, запустите программу SQLite Database Browser и откройте скачанную базу данных.Внимательно прочитайте задание, составьте запрос и выполните его в программе. Убедитесь, что результат запроса соответствует условиям задания. Если получился не тот результат, который вы ожидали, исправьте запрос и выполните его снова.

1. Составьте запрос, возвращающий все поля всех записей таблицы Categories.

2. Составьте запрос, возвращающий перечень названий всех товаров, по которым

ведётся детализированный учёт расходов.

3. Составьте запрос, возвращающий названия и описание всех категорий расхо-

дов.

4. Составьте запрос, возвращающий описание категории с номером 1.

5. Составьте запрос, возвращающий название и описание категории с номером 2.