

## Оглавление

Подготовка к изучению. Программы для работы с SQLite.....	6
Установка программы DB Browser for SQLite (для Windows и MacOS).....	6
Установка приложения aSQLiteManager (для Android).....	7
Как устроены таблицы баз данных.....	10
Записи.....	10
Поля таблиц.....	10
Типы полей.....	11
Связи между таблицами.....	11
Учебные базы данных.....	13
Учебная база данных №1. Списки дел.....	13
Учебная база данных №2. Учёт расходов.....	15
Запросы SELECT.....	20
Выбор нужных полей.....	21
Выбор нужных записей.....	22
Практика.....	24
Запросы к нескольким таблицам.....	25
Сокращение запросов с помощью псевдонимов.....	29
Практика.....	31
Условия отбора.....	32
Операторы сравнения.....	32
Отбор по спискам значений.....	33
Отбор по диапазонам значений.....	34
Комбинирование условий сравнения.....	35
Практика.....	37
Запрос с сортировкой.....	38
Сортировка по номерам полей вместо имён.....	40
Сортировка в обратном порядке.....	41
Практика.....	43
Удаление дубликатов.....	44
Практика.....	46
Добавление, изменение и удаление записей.....	47
Запросы INSERT.....	47
Запросы UPDATE.....	48
Запросы DELETE.....	49
Практика.....	50
Виды отношений.....	52
Отношения один-ко-многим.....	52

---

Отношения многие-ко-многим.....	53
Атрибуты отношения.....	54
Отношения один-к-одному.....	55
Практика.....	56
Создание таблиц.....	57
Типы данных ANSI SQL.....	59
Числовые данные.....	59
Символьные данные.....	62
Временные данные.....	63
Логические данные.....	64
Произвольные данные.....	64
Типы данных SQLite.....	64
Практика.....	66
NULL и значения по умолчанию.....	67
Удаление и изменение таблиц.....	70
Удаление таблицы.....	70
Изменение таблицы.....	71
Практика.....	73
Создание и использование представлений (VIEW).....	74
Создание и удаление представлений.....	74
Представления в запросах SELECT.....	76
Использование представлений.....	77
Практика.....	79
Ограничение числа возвращаемых записей.....	81
Практика.....	84
Вложенные запросы.....	85
SELECT в условиях отбора.....	85
Многоуровневые вложенные запросы.....	87
SELECT внутри INSERT.....	88
Практика.....	89
Функции и операции над полями.....	90
Использование функций в запросах SELECT.....	90
Группы функций.....	93
Математические функции и операции.....	93
Текстовые (строковые) функции.....	94
Функции даты и времени.....	96
Практика.....	100
Простой поиск по базе данных. Оператор LIKE.....	101
Ключевое слово ESCAPE.....	103

---

Практика.....	104
Способы комбинирования данных из нескольких таблиц. Ключевое слово JOIN.....	105
Практика.....	109
Группировка записей и агрегатные функции.....	110
Условия отбора групп. Ключевое слово HAVING.....	113
Использование агрегатных функций без группировки.....	115
Подсчёт общего количества записей. COUNT(*)......	116
Практика.....	117
Объединение результатов нескольких запросов. Использование UNION.....	118
Объединение с дублированными строками. UNION ALL.....	121
Практика.....	123
Что такое целостность данных.....	125
Ключи и индексы.....	126
Уникальные значения неключевых полей. UNIQUE.....	128
Практика.....	129
Проверка значений полей. Универсальное ограничение CHECK.....	130
Практика.....	131
Внешние ключи.....	132
Обработка связей при удалении и изменении. ON UPDATE и ON DELETE.....	133
Практика.....	135
Триггеры.....	138
Создание триггера на добавление записи.....	138
Условия срабатывания триггера.....	139
Тело триггера.....	140
Старые и новые значения полей.....	140
Разделитель запросов — точка с запятой.....	140
Триггер на удаление записи.....	141
Триггер на изменение записи.....	141
Практика.....	143
Так зачем же нужны программисты?.....	144
Причина первая. Графические интерфейсы.....	144
Причина вторая. Хранимые процедуры.....	146
Причина третья. СУБД.....	147
Получите сертификат!.....	148

# Подготовка к изучению. Программы для работы с SQLite

## Установка программы DB Browser for SQLite (для Windows и MacOS)

Цель нашего курса состоит в изучении основ языка SQL. Изучение будет активным: вы будете не просто читать тексты и отвечать на вопросы, а научитесь самостоятельно составлять запросы SQL и проверять результаты их выполнения на учебных базах данных.

Для того, чтобы это стало возможным, вам нужны файлы с этими учебными базами данных, а также программа, позволяющая вводить запросы SQL и получать результаты их выполнения. Для этого мы воспользуемся очень популярным *движком* для работы с базами данных: **SQLite**. На вашем компьютере наверняка уже установлена программа, использующая этот движок, и даже, может быть, не одна. Например, его используют Skype, браузер Firefox, Adobe Reader, а также множество других популярных программ.

SQLite представляет собой *встраиваемую* библиотеку для работы с базами данных. Это означает, что для использования SQLite не нужно отдельно устанавливать и настраивать специальные программные сервисы, которые требуются для большинства популярных типов БД.

Каждая база данных SQLite представляет собой всего один файл, а движок для работы с этой базой данных встраивается прямо в программу, которая её использует.

Файлы баз данных, которые мы будем использовать для изучения SQL, вы найдёте в уроках этого курса. Вам нужно будет их просто скачивать по мере выполнения уроков. А для работы с ними мы воспользуемся бесплатной программой DB Browser for SQLite, которую вы можете скачать с этого сайта: <http://sqlitebrowser.org>.

Эта программа позволяет делать всё, что нам понадобится в процессе изучения SQL. Она может:

- открывать разные файлы баз данных SQLite,
- показывать структуру базы данных и содержимое её таблиц
- и, конечно же, выполнять запросы к базе данных на языке SQL.

Скачайте и установите эту программу.

К сожалению, эта программа не портирована на Android. Поэтому, если вы планируете изучать этот курс на устройстве под управлением Android, то вам нужно будет использовать другую программу. Хорошим вариантом для этого является бесплатная программа aSQLiteManager, об установке и использовании которой рассказывается в следующей главе.

Всё, теперь мы можем начинать изучение языка SQL!

## Как устроены таблицы баз данных

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 этой ячейки 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

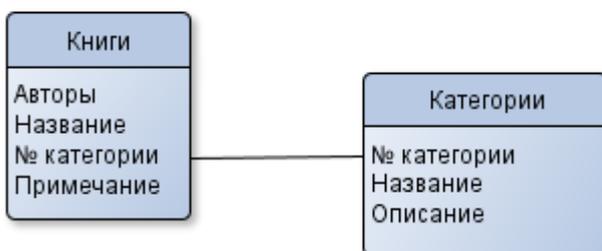
Обычно базы данных состоят из множества таблиц, записи которых могут быть связаны между собой. Чтобы понять, как они связываются, нам нужно перечитать самую первую фразу этой главы:

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.

## 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
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 сокращение 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 отдельный 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.

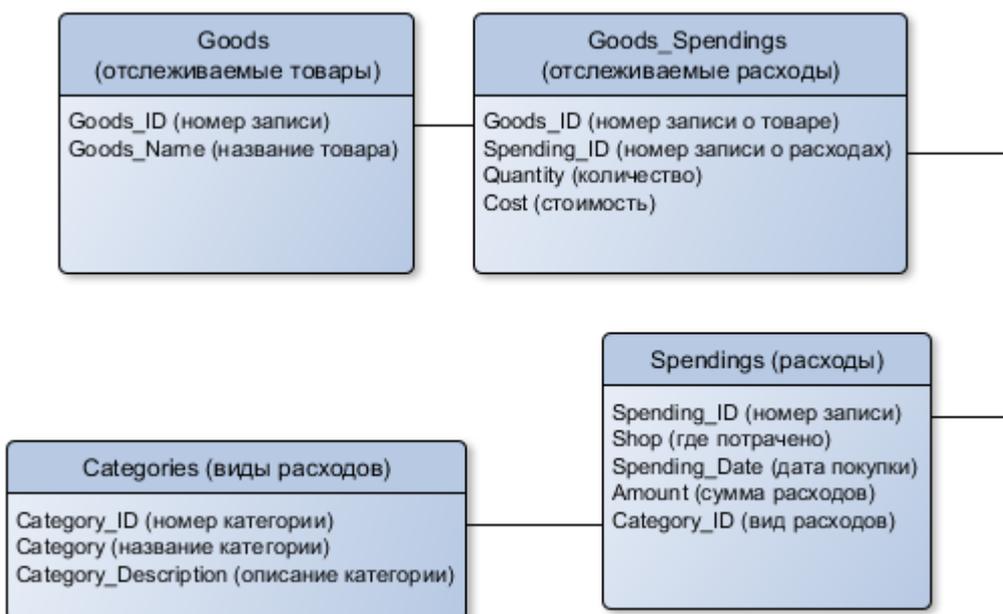


Таблица **Spending** (расходы) предназначена для учёта всех совершаемых расходов и включает такие поля.

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
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

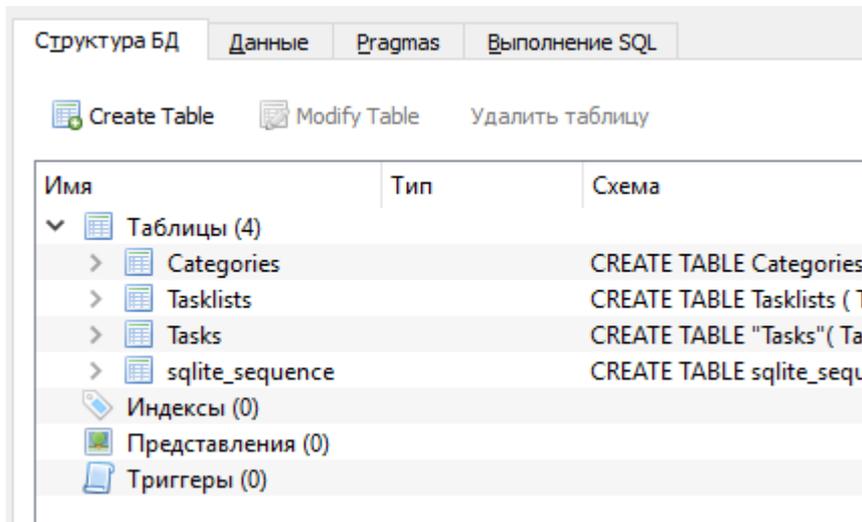
Наконец, таблица **Goods\_Spendings** предназначена для ведения дополнительного учёта по товарам из списка, представленного в таблице **Goods**. В таблице **Goods\_Spendings** сохраняется информация о количестве купленного товара, его стоимости, а также о покупке, в рамках которой был приобретён этот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

Поле	Описание	Тип поля
Goods_ID	Порядковый номер товара в таблице <b>Goods</b>	Целое число
Spending_ID	Порядковый номер покупки в таблице <b>Spendings</b> , 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

Скачайте файлы описанных здесь баз данных по этим ссылкам: [Списки дел](#), [Учёт расходов](#).

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.

Скачайте и 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.

	name	seq
	Filter	Filter
1	Tasks	11
2	Tasklists	4
3	Categories	5

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

Всё работает? Ну что же, всё готово для того, чтобы сейчас же написать и проверить первый запрос SQL!

# 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.

## 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.

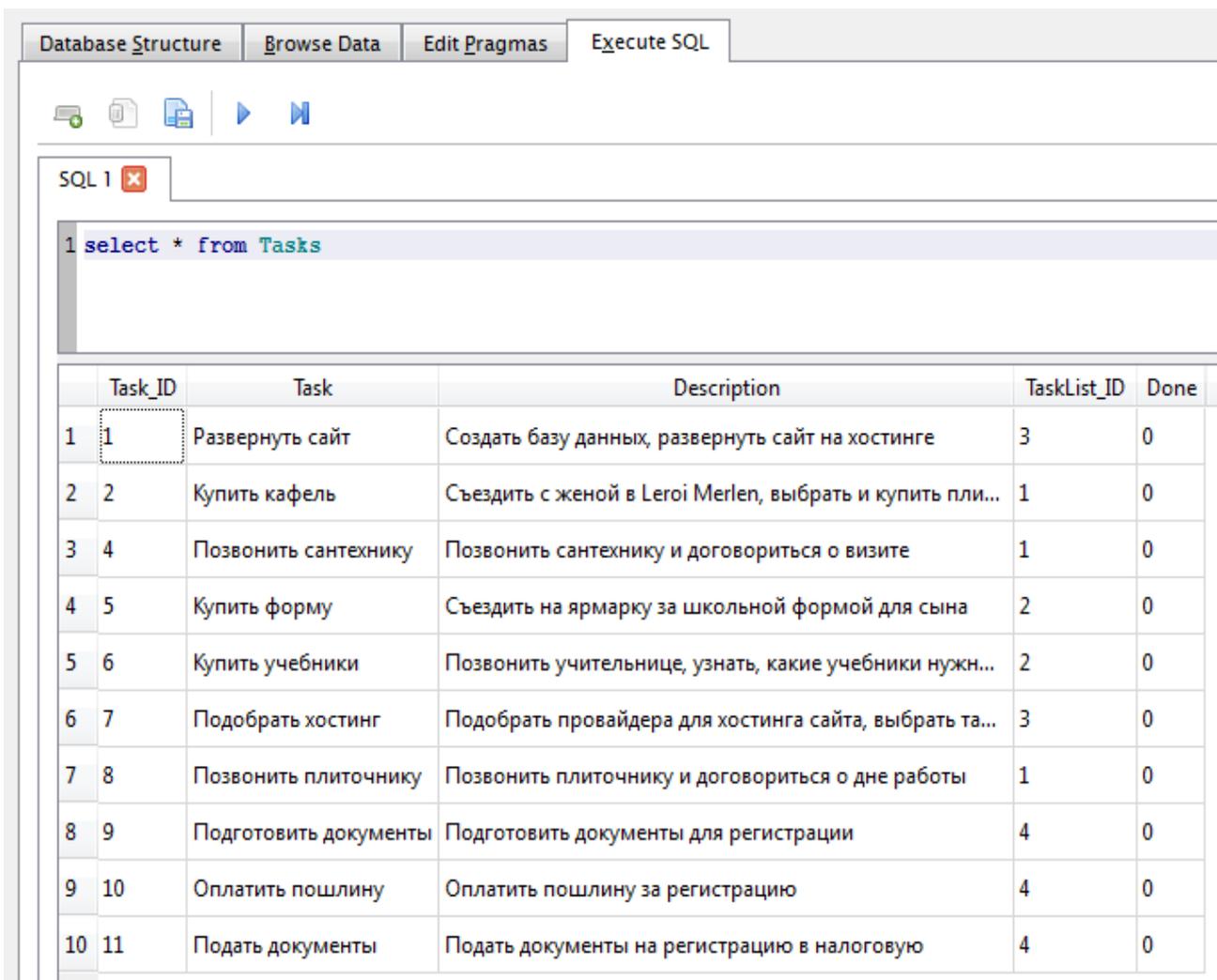
Запустите программу SQLite Database Browser и откройте нашу первую учебную базу данных.

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.



The screenshot shows the SQLite Database Browser interface. The 'Execute SQL' tab is active, displaying the query `select * from Tasks`. Below the query editor, a table with 6 columns is shown: Task\_ID, Task, Description, TaskList\_ID, and Done. The table contains 10 rows of data.

	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.

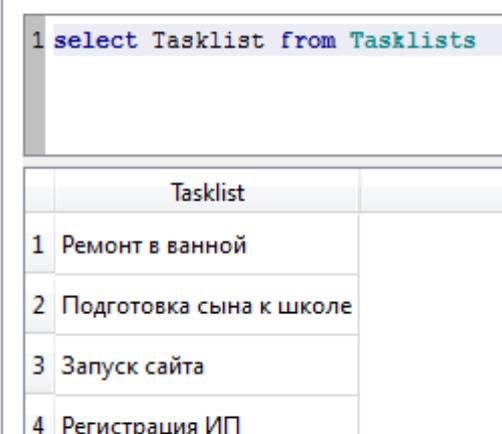
Вы уже поняли, как можно получить данные из других таблиц. Введите и выполните такие запросы, чтобы получить содержимое таблиц Tasklists и Categories:

```
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.

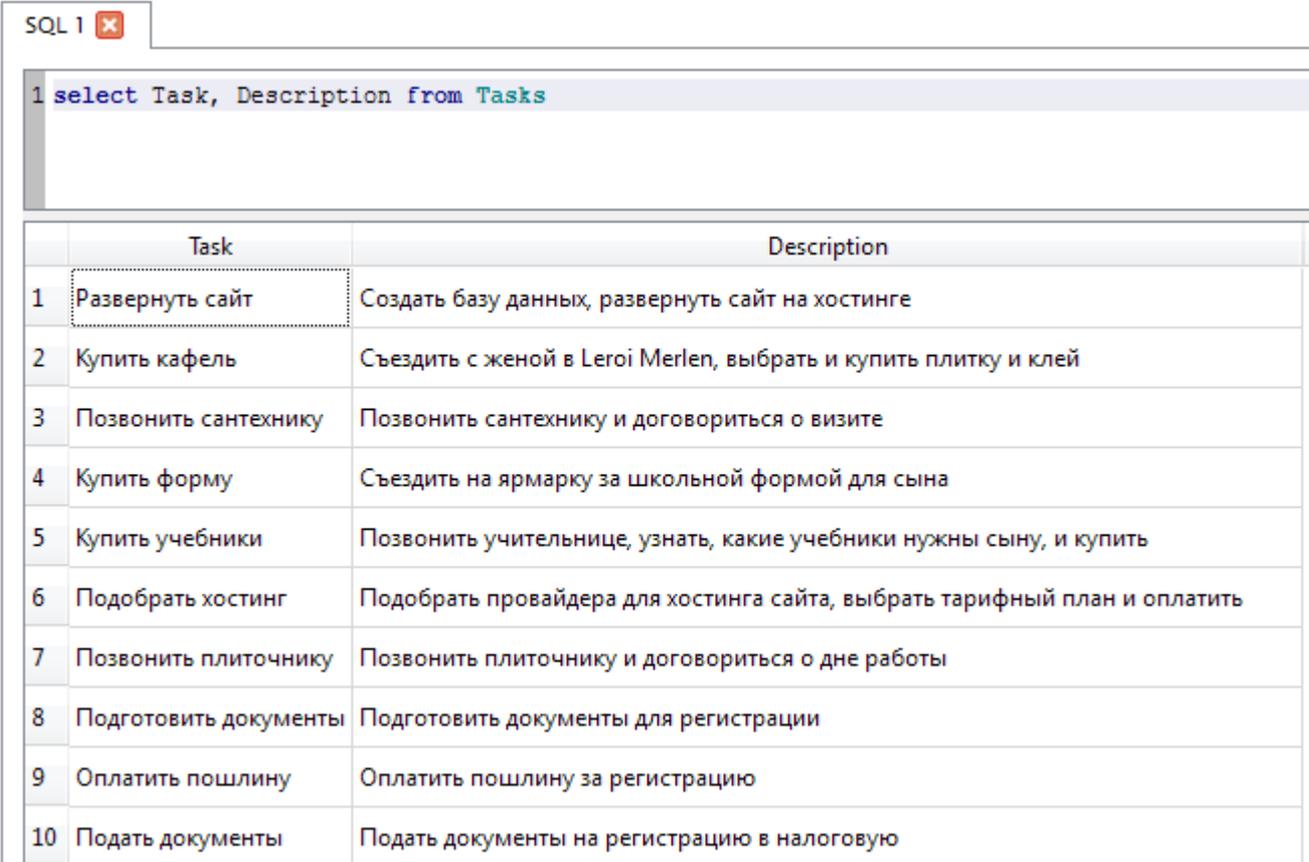
```
select Tasklist from Tasklists
```



	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
```



The screenshot shows a SQL IDE window titled "SQL 1" with a query editor containing the SQL statement: `1 select Task, Description from Tasks`. Below the editor, a table displays the results of the query. The table has two columns: "Task" and "Description". There are 10 rows of data, each with a task ID and a description.

	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
```

SQL 1 ✕

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

	Task	Description
1	Купить кафель	Съездить с женой в Leroi Merlen, выбрать и купить плитку и клей
2	Позвонить сантехнику	Позвонить сантехнику и договориться о визите
3	Позвонить плиточнику	Позвонить плиточнику и договориться о дне работы

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 отобразить 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
```

SQL 1 ✕

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

	Tasklist
1	Ремонт в ванной

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

Задания, приведенные в этой книге, соответствуют тестовым заданиям бесплатного онлайн-курса [«SQL для непрограммистов»](#). Вы можете воспользоваться соответствующими тестами курса, чтобы убедиться, что вы выполнили все задания правильно. При желании после выполнения всех тестовых заданий вы можете получить электронный сертификат об окончании курса (предоставляется за отдельную плату).

Для выполнения задания скачайте учебную базу данных [«Учёт расходов»](#), 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.

Ссылка на тест этого задания в курсе «SQL для непрограммистов»: [Запросы SELECT – тест](#)

- 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 расходов.*
- 4. Составьте запрос, возвращающий описание категории с номером 1.*
- 5. Составьте запрос, возвращающий название и описание категории с номером 2.*

## Запросы к нескольким таблицам

В предыдущей главе нам пришлось выполнить два запроса, чтобы

- а) увидеть все задачи, входящие в первый по номеру список и
- б) узнать название этого списка.

Так получилось, потому что к этому моменту мы знаем, как использовать запросы SELECT для выбора записей только из одной таблицы. Информация о задачах хранится в одной таблице (Tasks), а названия списков задач — в другой (Tasklists).

Можно ли объединить эти два запроса в один, обратившись сразу к двум таблицам? Конечно, можно! Введите и выполните такой запрос и посмотрите на результат:

```
select Tasklist, Task, Description
from Tasks, Tasklists
where Tasks.Tasklist_ID=Tasklists.Tasklist_ID
and Tasks.Tasklist_ID=1
```

SQL 1 ✖

```
1 select Tasklist, Task, Description
2 from Tasks, Tasklists
3 where Tasks.Tasklist_ID=Tasklists.Tasklist_ID
4 and Tasks.Tasklist_ID=1
```

	Tasklist	Task	Description
1	Ремонт в ванной	Купить кафель	Съездить с женой в Leroi Merlen, выбрать и купить плитку и клей
2	Ремонт в ванной	Позвонить сантехнику	Позвонить сантехнику и договориться о визите
3	Ремонт в ванной	Позвонить плиточнику	Позвонить плиточнику и договориться о дне работы

Теперь в результатах запроса мы видим и название списка задач, и сами задачи.

В этом запросе появилось три новых элемента. Давайте рассмотрим их подробнее.

Во-первых обобщённая структура запроса теперь выглядит так:

```
SELECT {список полей} FROM {список таблиц} WHERE {условие отбора}
```

Там, где мы раньше использовали имя одной таблицы, можно использовать целый список таблиц. Имена таблиц в этом списке разделяются запятой:

```
... from Tasks, Tasklists ...
```

Во-вторых, поскольку в двух наших таблицах присутствуют поля с одинаковыми именами Tasklist\_ID, каждый раз при использовании этого имени нам пришлось дополнительно сообщать движку базы данных, какую из таблиц мы имеем в виду. Для этого использовано расширенное имя поля, которое состоит из имени таблицы и собственно имени поля, разделённых точкой:

```
... where Tasks.Tasklist_ID = Tasklists.Tasklist_ID...
```

Если бы мы не указали принадлежность каждого из полей определённой таблице, движок вернул бы нам сообщение об ошибке примерно такого содержания: «не могу понять, к какой таблице относится поле Tasklist\_ID».

```
ambiguous column name: Tasklist id
```

В-третьих, в этом запросе мы использовали условие, составленное из двух проверок:

```
Tasks.Tasklist_ID=Tasklists.Tasklist_ID and Tasks.Tasklist_ID=1
```

Эти две проверки соединены ещё одним ключевым словом — **AND** (и). Оно означает, что для отбора записей должны быть выполнены оба условия:

- в таблицах Tasklists и Tasks должны совпадать значения полей Tasklist\_ID
- значение этого поля в таблице Tasks должно быть равно 1.

Этот запрос уже далеко не такой простой, как те, что мы использовали раньше. Чтобы лучше понять, как работают условия отбора, давайте попробуем выполнить его без условия **WHERE**:

```
select Tasklist, Task, Description
from Tasks, Tasklists
```

SQL 1 ✕

```
1 select Tasklist, Task, Description
2 from Tasks, Tasklists
3
```

	Tasklist	Task	Description
1	Ремонт в ванной	Развернуть сайт	Создать базу данных, развернуть сайт на хостинге
2	Подготовка сына к школе	Развернуть сайт	Создать базу данных, развернуть сайт на хостинге
3	Запуск сайта	Развернуть сайт	Создать базу данных, развернуть сайт на хостинге
4	Регистрация ИП	Развернуть сайт	Создать базу данных, развернуть сайт на хостинге
5	Ремонт в ванной	Купить кафель	Съездить с женой в Leroi Merlen, выбрать и купить ...
6	Подготовка сына к школе	Купить кафель	Съездить с женой в Leroi Merlen, выбрать и купить ...
7	Запуск сайта	Купить кафель	Съездить с женой в Leroi Merlen, выбрать и купить ...
8	Регистрация ИП	Купить кафель	Съездить с женой в Leroi Merlen, выбрать и купить ...

40 Rows returned from: select Tasklist, Task, Description  
 from Tasks, Tasklists (took 2ms)

Результат запроса состоит аж из 40 строк! И посмотрите: задача «Развернуть сайт» появилась в результатах запроса четыре раза, потому что при обработке запроса движок «включил» её в каждый из четырёх списков. То же самое произошло со всеми задачами. А поскольку списков у нас четыре, а задач десять, в результате и получилось  $4 \times 10 = 40$  строк.

Почему так получилось? Дело в том, что наша база данных на самом деле не знает, к какому списку относится какая задача. Хотя мы и дали одинаковые названия полям Tasklist\_ID в двух таблицах, это сделано только для нашего удобства. Имена полей могли быть и разными. Чтобы связать список задач в таблице Tasklists с задачами из таблицы Tasks, мы должны явно сообщить в запросе, что таблицы связаны именно по этим полям. Для этого и служит первая часть условия:

```
where Tasks.Tasklist_ID = Tasklists.Tasklist_ID
```

Если мы выполним запрос, добавив к нему только это условие, то движок базы данных отберёт из этих сорока строк только те комбинации записей таблиц Tasks и Tasklists, в которых значения полей Tasklist\_ID совпадают:

```
select Tasklist, Task, Description
from Tasks, Tasklists
where Tasks.Tasklist_ID=Tasklists.Tasklist_ID
```

SQL 1 

```
1 select Tasklist, Task, Description
2 from Tasks, Tasklists
3 where Tasks.Tasklist_ID=Tasklists.Tasklist_ID
4 |
```

	Tasklist	Task	Description
1	Запуск сайта	Развернуть сайт	Создать базу данных, развернуть сайт на хостинге
2	Ремонт в ванной	Купить кафель	Съездить с женой в Leroi Merlen, выбрать и купить ...
3	Ремонт в ванной	Позвонить сантехнику	Позвонить сантехнику и договориться о визите
4	Подготовка сына к школе	Купить форму	Съездить на ярмарку за школьной формой для сына
5	Подготовка сына к школе	Купить учебники	Позвонить учительнице, узнать, какие учебники ну...
6	Запуск сайта	Подобрать хостинг	Подобрать провайдера для хостинга сайта, выбрат...
7	Ремонт в ванной	Позвонить плиточнику	Позвонить плиточнику и договориться о дне работы
8	Регистрация ИП	Подготовить документы	Подготовить документы для регистрации
9	Регистрация ИП	Оплатить пошлину	Оплатить пошлину за регистрацию
10	Регистрация ИП	Подать документы	Подать документы на регистрацию в налоговую

```
10 Rows returned from: select Tasklist, Task, Description
from Tasks, Tasklists
where Tasks.Tasklist_ID=Tasklists.Tasklist_ID (took 4ms)
```

Теперь количество строк в результате запроса совпадает с количеством записей таблицы Tasks, и каждой задаче поставлен в соответствие только один список.

Обратите внимание: полю Tasklist\_ID, которое мы использовали в условии отбора, совсем не обязательно присутствовать в списке возвращаемых полей.

Что происходит, когда мы добавляем к запросу второе условие?

```
and Tasks.Tasklist_ID=1
```

Как вы, наверное, уже поняли, оно дополнительно отфильтровывает из результата предыдущего запроса только те строки, при формировании которых использовались записи таблицы Tasks, в которых значение поля Tasklist\_ID равно 1.

SQL 1 ✕

```

1 select Tasklist, Task, Description
2 from Tasks, Tasklists
3 where Tasks.Tasklist_ID=Tasklists.Tasklist_ID
4 and Tasks.Tasklist_ID=1

```

	Tasklist	Task	Description
1	Ремонт в ванной	Купить кафель	Съездить с женой в Leroi Merlen, выбрать и купить плитку и клей
2	Ремонт в ванной	Позвонить сантехнику	Позвонить сантехнику и договориться о визите
3	Ремонт в ванной	Позвонить плиточнику	Позвонить плиточнику и договориться о дне работы

В предыдущей главе я обещал вам, что можно избежать запоминания номеров записей в запросе. Наверное, вы и сами уже догадались, что в качестве условия отбора записей можно использовать поле **Tasklists.Tasklist** (название списка) вместо поля **Tasks.Tasklist\_ID** (номер списка):

```

select Tasklist, Task, Description
from Tasks, Tasklists
where Tasks.Tasklist_ID=Tasklists.Tasklist_ID
and Tasklists.Tasklist='Ремонт в ванной'

```

SQL 1 ✕

```

1 select Tasklist, Task, Description
2 from Tasks, Tasklists
3 where Tasks.Tasklist_ID=Tasklists.Tasklist_ID
4 and Tasklists.Tasklist='Ремонт в ванной'

```

	Tasklist	Task	Description
1	Ремонт в ванной	Купить кафель	Съездить с женой в Leroi Merlen, выбрать и купить плитку и клей
2	Ремонт в ванной	Позвонить сантехнику	Позвонить сантехнику и договориться о визите
3	Ремонт в ванной	Позвонить плиточнику	Позвонить плиточнику и договориться о дне работы

Если в запросе SQL используется строка, то она должна быть обрамлена одинарными кавычками:

```
and Tasklists.Tasklist='Ремонт в ванной'
```

Таблиц в списке может быть и больше двух, и в большинстве случаев каждая из перечисленных таблиц должна упоминаться хотя бы в одном условии, связывающем её с другими таблицами. Следующий запрос выбирает данные из трёх таблиц, добавляя к каждой задаче информацию о том, в какой список она входит, а также к какой категории дел относится этот список:

```
select Category, Tasklist, Task, Description
from Categories, Tasks, Tasklists
where Tasks.Tasklist_ID=Tasklists.Tasklist_ID
and Tasklists.Category_ID=Categories.Category_ID
```

SQL 1 ✕

```
1 select Category, Tasklist, Task, Description
2 from Categories, Tasks, Tasklists
3 where Tasks.Tasklist_ID=Tasklists.Tasklist_ID
4 and Tasklists.Category_ID=Categories.Category_ID
```

	Category	Tasklist	Task	Description
1	Своё дело	Запуск сайта	Развернуть сайт	Создать базу данных, развернуть сайт на хос
2	Дом	Ремонт в ванной	Купить кафель	Съездить с женой в Leroi Merlen, выбрать и
3	Дом	Ремонт в ванной	Позвонить сантехнику	Позвонить сантехнику и договориться о виз
4	Семья	Подготовка сына к школе	Купить форму	Съездить на ярмарку за школьной формой
5	Семья	Подготовка сына к школе	Купить учебники	Позвонить учительнице, узнать, какие учебн
6	Своё дело	Запуск сайта	Подобрать хостинг	Подобрать провайдера для хостинга сайта, и
7	Дом	Ремонт в ванной	Позвонить плиточнику	Позвонить плиточнику и договориться о дн
8	Своё дело	Регистрация ИП	Подготовить документы	Подготовить документы для регистрации
9	Своё дело	Регистрация ИП	Оплатить пошлину	Оплатить пошлину за регистрацию
10	Своё дело	Регистрация ИП	Подать документы	Подать документы на регистрацию в налогс

## Сокращение запросов с помощью псевдонимов

Изучаемые нами запросы становятся всё более громоздкими. Нам приходится по несколько раз включать в запрос названия таблиц, которые могут быть достаточно длинными. Это не очень удобно и увеличивает вероятность опечатки при вводе запроса.

В SQL есть способ сократить длинные запросы с помощью так называемых *псевдонимов*. Псевдоним таблицы представляет собой короткую замену названию таблицы, которая может использоваться в пределах запроса в полных именах полей.

Например, наш предыдущий запрос можно записать таким образом:

```
select Category, Tasklist, Task, Description
from Categories c, Tasks t, Tasklists tl
where t.Tasklist_ID=tl.Tasklist_ID
and tl.Category_ID=c.Category_ID
```

SQL 1 ✕

```
1 select Category, Tasklist, Task, Description
2 from Categories c, Tasks t, Tasklists tl
3 where t.Tasklist_ID=tl.Tasklist_ID
4 and tl.Category_ID=c.Category_ID
```

	Category	Tasklist	Task	Description
1	Своё дело	Запуск сайта	Развернуть сайт	Создать базу данных, развернуть сайт на хосты
2	Дом	Ремонт в ванной	Купить кафель	Съездить с женой в Leroi Merlen, выбрать и кут
3	Дом	Ремонт в ванной	Позвонить сантехнику	Позвонить сантехнику и договориться о визит
4	Семья	Подготовка сына к школе	Купить форму	Съездить на ярмарку за школьной формой дл
5	Семья	Подготовка сына к школе	Купить учебники	Позвонить учительнице, узнать, какие учебни
6	Своё дело	Запуск сайта	Подобрать хостинг	Подобрать провайдера для хостинга сайта, вы
7	Дом	Ремонт в ванной	Позвонить плиточнику	Позвонить плиточнику и договориться о дне р
8	Своё дело	Регистрация ИП	Подготовить документы	Подготовить документы для регистрации
9	Своё дело	Регистрация ИП	Оплатить пошлину	Оплатить пошлину за регистрацию
10	Своё дело	Регистрация ИП	Подать документы	Подать документы на регистрацию в налогову

Для каждой таблицы в этом запросе определён псевдоним. Псевдоним указывается сразу после настоящего имени таблицы в конструкции FROM:

```
from Categories c, Tasks t, Tasklists tl
```

В нашем примере псевдонимы использовались для сокращения текста запроса в условиях отбора после ключевого слова WHERE. Но их точно так же можно использовать и до ключевого слова FROM в полных именах полей, например:

```
select tl.Category_ID, Tasklist
from Categories c, Tasklists tl
where tl.Category_ID=c.Category_ID
```

## Практика

Для выполнения задания скачайте учебную базу данных [«Учёт расходов»](#), запустите программу SQLite Database Browser и откройте скачанную базу данных.

Ссылка на тест этого задания в курсе «SQL для непрограммистов»:

[Запросы к нескольким таблицам — тест](#)

1. Составьте запрос, выбирающий из таблиц *Goods* и *Goods\_Spending* и возвращающий перечень названий отслеживаемых товаров и всех потраченных на них сумм.
2. Составьте запрос к таблицам *Categories* и *Spending*s, который возвращает название категории покупки, название магазина и потраченную сумму для всех покупок, относящихся к категории с номером 4.
3. Составьте запрос к таблицам *Goods* и *Goods\_Spendings*, возвращающий наименование товара и количество купленных единиц по всем покупкам для товара «Пиво».
4. Доработайте предыдущий запрос, добавив к результатам дату покупки из таблицы *Spending*s. Таким образом, ваш запрос должен обращаться к трём таблицам и возвращать наименование товара, количество купленных единиц и дату каждой покупки для товара «Пиво».
5. Составьте запрос, обращающийся ко всем четырём таблицам (*Categories*, *Spending*s, *Goods*, *Goods\_Spendings*), возвращающий следующую информацию обо всех покупках отслеживаемых товаров: название категории, название товара, название магазина и дату покупки.

## Условия отбора

### Операторы сравнения

В рассмотренных примерах мы до сих пор использовали в качестве условий отбора записей только выражения, содержащие знак равенства, например:

```
Tasklists.Tasklist='Ремонт в ванной'
```

Это выражение можно словами описать таким образом: «значение поля Tasklist таблицы Tasklists совпадает со строкой 'Ремонт в ванной'».

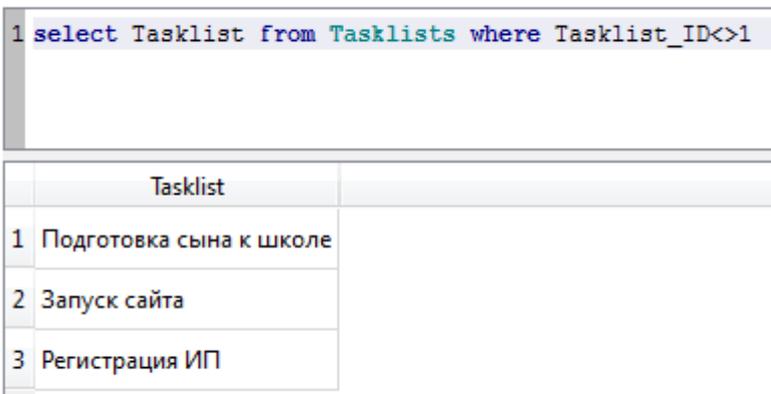
Проверка на совпадение значений — это, конечно, не единственное допустимое выражение в условиях отбора. SQL позволяет использовать в условиях разнообразные операторы сравнения. К числу самых распространённых относятся следующие:

= — уже знакомый нам оператор сравнения на совпадение или равенство;

<> или != — оператор сравнения на несовпадение или неравенство. Результат сравнения противоположен результату условия с оператором =

Например, следующий запрос выберет из таблицы Tasklists названия всех списков задач, *кроме* списка с номером 1:

```
select Tasklist from Tasklists where Tasklist_ID<>1
```



The image shows a screenshot of a SQL query and its result set. The query is: `1 select Tasklist from Tasklists where Tasklist_ID<>1`. The result set is a table with two columns: an implicit ID column and a 'Tasklist' column. The rows are:

	Tasklist
1	Подготовка сына к школе
2	Запуск сайта
3	Регистрация ИП

< , > , <= , >= - меньше, больше, меньше или равно, больше или равно.

Результат сравнения с этими операторами зависит от типа данных. Для численных данных операторы сравнения работают обычным, знакомым нам со школьных лет способом. Например, следующий запрос выберет из таблицы Tasks все записи с номером задачи больше 5.

```
select * from Tasks where Task_ID>5
```

```
1 select * from Tasks where Task_ID>5|
```

	Task_ID	Task	Description	TaskList_ID	Done
1	6	Купить учебники	Позвонить учительнице, узнать, какие учебники ну...	2	0
2	7	Подобрать хостинг	Подобрать провайдера для хостинга сайта, выбрат...	3	0
3	8	Позвонить плиточнику	Позвонить плиточнику и договориться о дне работы	1	0
4	9	Подготовить документы	Подготовить документы для регистрации	4	0
5	10	Оплатить пошлину	Оплатить пошлину за регистрацию	4	1
6	11	Подать документы	Подать документы на регистрацию в налоговую	4	0

Для данных типов «дата» и «время» эти операторы работают интуитивно понятным образом, используя хронологический порядок: `<` означает «раньше», `>` — «позже». `<=` и `>=`, соответственно, означают «не позже» и «не раньше».

Для данных типа «текст» производится побуквенное сравнение, начиная с первой буквы. При этом используется алфавитный порядок: выполнение условия `'a' < 'б'` означает, что левая буква находится ближе к началу алфавита, чем правая. Но надо сказать, что результат сравнения зависит от типа и языковых настроек используемой базы данных. В частности, заглавные и строчные буквы могут считаться двумя разными наборами букв, так что их сравнение оказывается бессмысленным. Например, может выполняться такое условие: `'a' > 'Б'`. У каждого типа баз данных есть собственные настройки таких сравнений для текстовых данных. Если вы не знаете особенностей этих настроек, то лучше избегать сравнения текстовых строк с использованием этих операторов, потому что результат может оказаться совсем не таким, как вы ожидали.

Операторы сравнения удобно использовать, когда мы имеем дело с количественными значениями. В нашей учебной базе «Список дел» числа используются только как идентификаторы или признаки, поэтому условие вроде `Task_ID>5` выглядит оторванным от жизни. Более осмысленные задачи с использованием операторов сравнения вы будете решать со второй учебной базой, «Учёт расходов», в которой хранятся данные о суммах покупок.

## Отбор по спискам значений

В условии отбора можно указать список возможных значений поля. Для этого используется ключевое слово `IN`, после которого в скобках перечисляются значения. Значения разделяются запятыми.

Следующий запрос выберет из таблицы **Tasks** записи задач, которые относятся к спискам №№ 2, 3 и 4.

```
select * from Tasks where Tasklist_ID in (2,3,4)
```

```
1 select * from Tasks where Tasklist_ID in (2,3,4)
```

	Task_ID	Task	Description	TaskList_ID	Done
1	1	Развернуть сайт	Создать базу данных, развернуть сайт на хостинге	3	0
2	5	Купить форму	Съездить на ярмарку за школьной формой для сына	2	0
3	6	Купить учебники	Позвонить учительнице, узнать, какие учебники ну...	2	0
4	7	Подобрать хостинг	Подобрать провайдера для хостинга сайта, выбрат...	3	0
5	9	Подготовить документы	Подготовить документы для регистрации	4	0
6	10	Оплатить пошлину	Оплатить пошлину за регистрацию	4	1
7	11	Подать документы	Подать документы на регистрацию в налоговую	4	0

Выражение IN с перечислением можно использовать не только с числами, а с любыми типами значений. Например, со строками:

```
select * from Categories where Category in ('Семья', 'Дом')
```

```
1 select * from Categories where Category in ('Семья', 'Дом')
```

	Category_ID	Category	Category_Description
1	1	Семья	Семейные дела.
2	3	Дом	Дела по дому: покупки, ремонт, оплата счетов.

## Отбор по диапазонам значений

В SQL также есть способ указать диапазон значений в условии отбора. Для этого используется ключевое слово BETWEEN, после которого указываются минимальное и максимальное значения, разделённые словом AND. Следующий запрос тоже отберёт из таблицы Tasks записи задач из списков №№2, 3 и 4, но вместо перечисления мы укажем в нём диапазон значений:

```
select * from Tasks where Tasklist_ID between 2 and 4
```

```
1 select * from Tasks where Tasklist_ID between 2 and 4
2
```

	Task_ID	Task	Description	TaskList_ID	Done
1	1	Развернуть сайт	Создать базу данных, развернуть сайт на хостинге	3	0
2	5	Купить форму	Съездить на ярмарку за школьной формой для сына	2	0
3	6	Купить учебники	Позвонить учительнице, узнать, какие учебники ну...	2	0
4	7	Подобрать хостинг	Подобрать провайдера для хостинга сайта, выбрат...	3	0
5	9	Подготовить документы	Подготовить документы для регистрации	4	0
6	10	Оплатить пошлину	Оплатить пошлину за регистрацию	4	1
7	11	Подать документы	Подать документы на регистрацию в налоговую	4	0

## Комбинирование условий сравнения

Мы уже использовали комбинацию условий с использованием ключевого слова AND в запросе, выбирающем из двух таблиц данные по задачам, относящимся к списку с номером 1:

```
select Tasklist, Task, Description from Tasks, Tasklists
where Tasks.Tasklist_ID=Tasklists.Tasklist_ID
and Tasks.Tasklist_ID=1
```

Ключевое слово AND (и) в данном случае означает, что должны выполняться оба условия, которые оно объединяет.

SQL поддерживает также ключевое слово OR (или). Оно означает, что должно выполняться хотя бы одно из указанных условий. Например, следующий запрос выберет информацию о задачах, входящих в список задач номер 3 или номер 4.

```
select * from Tasks
where Tasklist_ID=3 or Tasklist_ID=4
```

```
1 select * from Tasks
2 where Tasklist_ID=3 or Tasklist_ID=4
```

	Task_ID	Task	Description	TaskList_ID	Done
1	1	Развернуть сайт	Создать базу данных, развернуть сайт на хостинге	3	0
2	7	Подобрать хостинг	Подобрать провайдера для хостинга сайта, выбрат...	3	0
3	9	Подготовить документы	Подготовить документы для регистрации	4	0
4	10	Оплатить пошлину	Оплатить пошлину за регистрацию	4	1
5	11	Подать документы	Подать документы на регистрацию в налоговую	4	0

В условиях отбора можно комбинировать сколько угодно выражений сравнения, объединяемых ключевыми словами AND и OR. Но при этом нужно очень хорошо представлять, в каком порядке будут выполняться эти сравнения. Проблема в том, что при сравнении ключевые слова AND и OR имеют разный приоритет: сначала обрабатываются все условия, связанные словом AND, и только после этого — комбинации условий с помощью OR.

Например, если мы хотим отобразить записи по всем выполненным задачам (Done = 1), входящим в списки №3 и №4, то следующий запрос вернёт неправильный результат:

```
select * from Tasks where Tasklist_ID=3 or Tasklist_ID=4 and Done=1
```

```
1 select * from Tasks where Tasklist_ID=3 or Tasklist_ID=4 and Done=1
```

	Task_ID	Task	Description	TaskList_ID	Done
1	1	Развернуть сайт	Создать базу данных, развернуть сайт на хостинге	3	0
2	7	Подобрать хостинг	Подобрать провайдера для хостинга сайта, выбрат...	3	0
3	10	Оплатить пошлину	Оплатить пошлину за регистрацию	4	1

Почему так получилось? Потому что при разборе запроса движок сначала объединил условия, связанные словом AND:

```
Tasklist_ID=4 and Done=1
```

А потом добавил условие, стоящее перед OR:

```
Tasklist_ID=3
```

Полученную комбинацию условий можно словами описать так: «задача входит в список №4 и выполнена, либо задача входит в список №3 (и при этом неважно, выполнена или нет)».

Чтобы избежать подобной путаницы, можно использовать скобки. Комбинации условий в скобках обрабатываются с наивысшим приоритетом:

```
select * from Tasks where (Tasklist_ID=3 or Tasklist_ID=4) and Done=1
```

```
1 select * from Tasks where (Tasklist_ID=3 or Tasklist_ID=4) and Done=1
```

	Task_ID	Task	Description	TaskList_ID	Done
1	10	Оплатить пошлину	Оплатить пошлину за регистрацию	4	1

Для этого примера есть ещё лучший способ избежать путаницы. Если указать диапазон значений в условии отбора по полю Tasklist\_ID, то неоднозначность пропадает:

```
select * from Tasks where Tasklist_ID in (3,4) and Done=1
```

Это хорошая практика составления удобочитаемых запросов: использовать условия с ключевыми словами IN (задавая перечень значений) или BETWEEN (указывая диапазон значений) вместо комбинирования условий с помощью OR. Кроме того, в большинстве случаев такие запросы ещё и быстрее выполняются.

## Практика

Для выполнения задания скачайте учебную базу данных [«Учёт расходов»](#), запустите программу SQLite Database Browser и откройте скачанную базу данных.

Ссылка на тест этого задания в курсе «SQL для непрограммистов»: [Условия отбора — тест](#)

1. Составьте запрос, возвращающий значения всех полей таблицы *Spending*s для покупок на сумму более 1000 руб.
2. Составьте запрос к таблице *Spending*s, возвращающий название магазина и потраченную сумму для покупок, совершённых не позже 3 июня 2015 года.  
*Подсказка. Эту дату в запросе нужно указать в виде такой текстовой строки: '2015-06-03'*
3. Составьте запрос, возвращающий названия категорий с номерами 1, 3 и 4 (с использованием ключевого слова *IN*)
4. Составьте запрос к таблице *Spending*s, возвращающий название магазина, дату покупки и сумму для покупок, сумма которых находится в диапазоне от 1000 до 2000 руб (с использованием ключевого слова *between*).
5. Составьте запрос, обращающийся к таблицам *Goods* и *Goods\_Spending*s и возвращающий названия отслеживаемых товаров и потраченные на них суммы, превышающие 500 руб.

## Запрос с сортировкой

В предыдущем примере мы получили перечень всех задач с указанием списка, в который они входят, а также категории, к которой относится этот список.

SQL 1 ✕

```

1 select Category, Tasklist, Task, Description
2 from Categories, Tasks, Tasklists
3 where Tasks.Tasklist_ID=Tasklists.Tasklist_ID
4 and Tasklists.Category_ID=Categories.Category_ID

```

	Category	Tasklist	Task	Description
1	Своё дело	Запуск сайта	Развернуть сайт	Создать базу данных, развернуть сайт на хос
2	Дом	Ремонт в ванной	Купить кафель	Съездить с женой в Leroi Merlen, выбрать и
3	Дом	Ремонт в ванной	Позвонить сантехнику	Позвонить сантехнику и договориться о виз
4	Семья	Подготовка сына к школе	Купить форму	Съездить на ярмарку за школьной формой
5	Семья	Подготовка сына к школе	Купить учебники	Позвонить учительнице, узнать, какие учебн
6	Своё дело	Запуск сайта	Подобрать хостинг	Подобрать провайдера для хостинга сайта, и
7	Дом	Ремонт в ванной	Позвонить плиточнику	Позвонить плиточнику и договориться о дн
8	Своё дело	Регистрация ИП	Подготовить документы	Подготовить документы для регистрации
9	Своё дело	Регистрация ИП	Оплатить пошлину	Оплатить пошлину за регистрацию
10	Своё дело	Регистрация ИП	Подать документы	Подать документы на регистрацию в налогс

Но результат запроса выглядит не очень красиво, и его неудобно читать. Категории и списки «размазаны» по строкам, и хочется их сгруппировать, чтобы работать со списком было удобнее.

SQL предоставляет такую возможность с помощью ещё одного ключевого слова: **ORDER BY** (сортировать по). Следующий запрос отсортирует результаты по названию списка задач.

```

select Category, Tasklist, Task, Description
from Categories, Tasks, Tasklists
where Tasks.Tasklist_ID=Tasklists.Tasklist_ID
and Tasklists.Category_ID=Categories.Category_ID
order by Tasklist

```

SQL 1

```

1 select Category, Tasklist, Task, Description
2 from Categories, Tasks, Tasklists
3 where Tasks.Tasklist_ID=Tasklists.Tasklist_ID
4 and Tasklists.Category_ID=Categories.Category_ID
5 order by Tasklist

```

	Category	Tasklist	Task	Description
1	Своё дело	Запуск сайта	Развернуть сайт	Создать базу данных, развернуть сайт на хос
2	Своё дело	Запуск сайта	Подобрать хостинг	Подобрать провайдера для хостинга сайта, и
3	Семья	Подготовка сына к школе	Купить форму	Съездить на ярмарку за школьной формой
4	Семья	Подготовка сына к школе	Купить учебники	Позвонить учительнице, узнать, какие учебн
5	Своё дело	Регистрация ИП	Подготовить документы	Подготовить документы для регистрации
6	Своё дело	Регистрация ИП	Оплатить пошлину	Оплатить пошлину за регистрацию
7	Своё дело	Регистрация ИП	Подать документы	Подать документы на регистрацию в налогс
8	Дом	Ремонт в ванной	Купить кафель	Съездить с женой в Leroi Merlen, выбрать и
9	Дом	Ремонт в ванной	Позвонить сантехнику	Позвонить сантехнику и договориться о виз
10	Дом	Ремонт в ванной	Позвонить плиточнику	Позвонить плиточнику и договориться о дн

В обобщённом виде запрос теперь выглядит так:

```
SELECT {список полей} FROM {список таблиц} WHERE {условие отбора} ORDER BY
{список полей}
```

Список полей для сортировки может включать несколько полей. При этом результаты запроса сначала сортируются по первому полю в списке. Строки, в которых значения первого поля одинаковы, дополнительно сортируются по второму полю, и так далее, пока не будет выполнена сортировка по всем перечисленным полям.

Например, мы можем отсортировать результаты сначала по названию категории, а внутри этой сортировки — по названию списка задач. Запрос будет выглядеть так:

```
select Category, Tasklist, Task, Description
from Categories, Tasks, Tasklists
where Tasks.Tasklist_ID=Tasklists.Tasklist_ID
and Tasklists.Category_ID=Categories.Category_ID
order by Category, Tasklist
```

SQL 1

```

1 select Category, Tasklist, Task, Description
2 from Categories, Tasks, Tasklists
3 where Tasks.Tasklist_ID=Tasklists.Tasklist_ID
4 and Tasklists.Category_ID=Categories.Category_ID
5 order by Category, Tasklist

```

	Category	Tasklist	Task	Description
1	Дом	Ремонт в ванной	Купить кафель	Съездить с женой в Leroi Merlen, выбрать и к
2	Дом	Ремонт в ванной	Позвонить сантехнику	Позвонить сантехнику и договориться о визи
3	Дом	Ремонт в ванной	Позвонить плиточнику	Позвонить плиточнику и договориться о дне
4	Своё дело	Запуск сайта	Развернуть сайт	Создать базу данных, развернуть сайт на хост
5	Своё дело	Запуск сайта	Подобрать хостинг	Подобрать провайдера для хостинга сайта, в
6	Своё дело	Регистрация ИП	Подготовить документы	Подготовить документы для регистрации
7	Своё дело	Регистрация ИП	Оплатить пошлину	Оплатить пошлину за регистрацию
8	Своё дело	Регистрация ИП	Подать документы	Подать документы на регистрацию в налогот
9	Семья	Подготовка сына к школе	Купить форму	Съездить на ярмарку за школьной формой д
10	Семья	Подготовка сына к школе	Купить учебники	Позвонить учительнице, узнать, какие учебн

## Сортировка по номерам полей вместо имён

В списке полей вместо названий полей можно использовать их порядковые номера в результатах запроса. В предыдущем примере вместо

```
order by Category, Tasklist
```

можно написать:

```
order by 1, 2
```

Весь запрос тогда будет выглядеть таким образом:

```

select Category, Tasklist, Task, Description
from Categories, Tasks, Tasklists
where Tasks.Tasklist_ID=Tasklists.Tasklist_ID
and Tasklists.Category_ID=Categories.Category_ID
order by 1, 2

```

Результат его работы не изменится.

SQL 1 

```

1 select Category, Tasklist, Task, Description
2 from Categories, Tasks, Tasklists
3 where Tasks.Tasklist_ID=Tasklists.Tasklist_ID
4 and Tasklists.Category_ID=Categories.Category_ID
5 order by 1,2

```

	Category	Tasklist	Task	Description
1	Дом	Ремонт в ванной	Купить кафель	Съездить с женой в Leroi Merlen, выбрать и к
2	Дом	Ремонт в ванной	Позвонить сантехнику	Позвонить сантехнику и договориться о визи
3	Дом	Ремонт в ванной	Позвонить плиточнику	Позвонить плиточнику и договориться о дне
4	Своё дело	Запуск сайта	Развернуть сайт	Создать базу данных, развернуть сайт на хост
5	Своё дело	Запуск сайта	Подобрать хостинг	Подобрать провайдера для хостинга сайта, в
6	Своё дело	Регистрация ИП	Подготовить документы	Подготовить документы для регистрации
7	Своё дело	Регистрация ИП	Оплатить пошлину	Оплатить пошлину за регистрацию
8	Своё дело	Регистрация ИП	Подать документы	Подать документы на регистрацию в налого
9	Семья	Подготовка сына к школе	Купить форму	Съездить на ярмарку за школьной формой д
10	Семья	Подготовка сына к школе	Купить учебники	Позвонить учительнице, узнать, какие учебн

## Сортировка в обратном порядке

По умолчанию ORDER BY сортирует числовые значения по возрастанию, а текстовые поля по алфавиту. Но этот порядок можно изменить на обратный с помощью ключевого слова DESC, которое добавляется после имени (или номера) поля, по которому выполняется сортировка.

Сравните результат выполнения следующих двух запросов. Первый запрос выбирает из таблицы **Tasks** все краткие описания задач и сортирует их по алфавиту в порядке возрастания.

```
select task from Tasks order by Task
```

Второй запрос делает то же самое (только вместо имени поля указан его номер), но в выражение ORDER BY добавлено слово DESC.

```
select Task from Tasks order by 1 desc
```

SQL 1	
1 select Task from Tasks order by Task	
Task	
1	Купить кафель
2	Купить учебники
3	Купить форму
4	Оплатить пошлину
5	Подать документы
6	Подготовить документы
7	Подобрать хостинг
8	Позвонить плиточнику
9	Позвонить сантехнику
10	Развернуть сайт

SQL 1	
1 select Task from Tasks order by 1 desc	
Task	
1	Развернуть сайт
2	Позвонить сантехнику
3	Позвонить плиточнику
4	Подобрать хостинг
5	Подготовить документы
6	Подать документы
7	Оплатить пошлину
8	Купить форму
9	Купить учебники
10	Купить кафель

Противоположностью ключевого слова DESC является ключевое слово ASC. Эти слова являются сокращениями от английских слов **ascending** (восходящий) и **descending** (нисходящий).

Слово ASC обычно не указывают, потому что порядок сортировки по возрастанию используется по умолчанию.

Если используется сортировка по нескольким полям, то слова ASC и DESC относятся только к тому полю, после которого они указаны в запросе. Сравните результаты следующих двух запросов, которые отличаются только позицией, в которой стоит слово DESC.

Оба запроса выбирают из таблиц Tasklists и Tasks название списка задач (поле **Tasklist**) и краткое описание задачи (поле **Task**). В обоих запросах для сортировки используются последовательно поля **Tasklist** и **Task**. Но в первом запросе слово DESC стоит после Tasklist, а во втором — после Task.

```
select Tasklist, Task
from Tasklists, Tasks
where Tasklists.Tasklist_ID=Tasks.TaskList_ID
order by Tasklist desc, Task
```

```
select Tasklist, Task
from Tasklists, Tasks
where Tasklists.Tasklist_ID=Tasks.TaskList_ID
order by Tasklist, Task desc
```

	Tasklist	Task		Tasklist	Task
1	Ремонт в ванной	Купить кафель	1	Запуск сайта	Развернуть сайт
2	Ремонт в ванной	Позвонить плиточнику	2	Запуск сайта	Подобрать хостинг
3	Ремонт в ванной	Позвонить сантехнику	3	Подготовка сына к школе	Купить форму
4	Регистрация ИП	Оплатить пошлину	4	Подготовка сына к школе	Купить учебники
5	Регистрация ИП	Подать документы	5	Регистрация ИП	Подготовить документ
6	Регистрация ИП	Подготовить документы	6	Регистрация ИП	Подать документы
7	Подготовка сына к школе	Купить учебники	7	Регистрация ИП	Оплатить пошлину
8	Подготовка сына к школе	Купить форму	8	Ремонт в ванной	Позвонить сантехнику
9	Запуск сайта	Подобрать хостинг	9	Ремонт в ванной	Позвонить плиточнику
10	Запуск сайта	Развернуть сайт	10	Ремонт в ванной	Купить кафель

## Практика

Для выполнения задания скачайте учебную базу данных [«Учёт расходов»](#), запустите программу SQLite Database Browser и откройте скачанную базу данных.

Ссылка на тест этого и следующего задания в курсе «SQL для непрограммистов»:

[Сортировка и удаление дубликатов — тест](#)

1. Составьте запрос, возвращающий названия всех категорий покупок, отсортированных по алфавиту.
2. Составьте запрос к таблице *Spending*s, возвращающий названия магазинов, даты совершенных покупок и суммы покупок, отсортированные в порядке убывания сумм.
3. Составьте запрос, возвращающий названия отслеживаемых товаров и все потраченные на них суммы в следующем порядке: названия отсортированы по алфавиту, а суммы по каждому товару отсортированы в порядке уменьшения.
4. Составьте запрос, возвращающий название отслеживаемого товара, название магазина, в котором он был куплен и потраченную на этот товар сумму для тех покупок, в которых эта сумма превышает 200 руб, отсортированные в следующем порядке: название товара по алфавиту, название магазина по алфавиту, потраченная сумма в порядке уменьшения.

Это демонстрационный фрагмент книги «SQL для непрограммистов». Полная версия книги доступна на этой странице:

<https://www.webursitet.ru/product/sql-dlya-neprogrammistov-kniga.html>