# 1 Знакомство с базами данных и SQL

**В этой главе**

- ✓ Фундамент для дальнейшей работы
- ✓ Основы реляционных баз данных
- ✓ Обзор проектирования баз данных
- ✓ Основные понятия языка SQL для написания первого SQL-запроса

**Необходимые знания**

В этой главе вам встретятся фрагменты кода. Если вы захотите их выполнить или изучить их варианты для разных систем управления базами данных (СУБД), откройте репозиторий GitHub к этой книге (https://github.com/Neo-Hao/grokking-relational-database-design). Скрипты из этой главы находятся в папке `chapter_01`, а инструкции по их запуску — в файле `README.md`.

# Введение

Проектирование БД — критически важный этап программной разработки, которым, однако, часто пренебрегают. Практически каждое приложение должно где-то хранить данные и как-то управлять ими, но далеко не все могут похвастаться хорошо спроектированной базой данных. Подходя к этой задаче без понимания принципов эффективного построения баз данных, можно столкнуться с неожиданными проблемами, в том числе с беспорядочной организацией информации или задержками при выполнении запросов, которые требуют слишком много ресурсов. Все это может привести к неудобству для пользователей и багам.

Напротив, хорошо спроектированная база данных станет прочной основой для разработки эффективных приложений: грамотная организация и структура данных облегчают выборку и другие операции с ними, а значит, снижается количество ошибок и пользователям становится более комфортно работать.

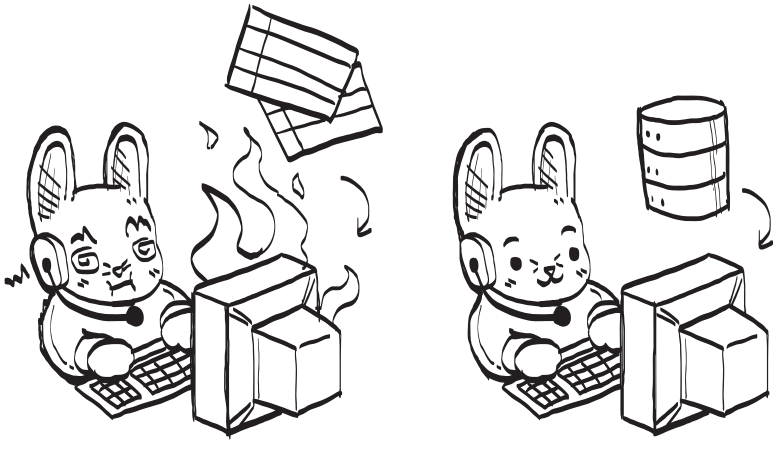
Каким бы ни был ваш уровень как программиста и разработчика приложений, важно уметь создавать эффективные базы данных и объяснять принципы их работы людям, возможно, далеким от технологий, не вводя их в ступор. Поможет вам в этом книга, которую вы держите в руках и где вы найдете многочисленные примеры, определения и пояснения, написанные простым языком. Даже при нулевых начальных

познаниях, перелистнув последнюю страницу, вы будете хорошо разбираться в том, как строятся реляционные базы данных.

А в этой главе мы познакомимся с ними, определим несколько ключевых понятий, которые будем использовать в дальнейшем, а также изучим основы *языка структурированных запросов* (Structured Query Language, SQL). SQL — язык программирования, который предназначен для работы с информацией, хранящейся в реляционных базах данных. Для их разработки важно уверенно владеть этим языком.

## Реляционные базы данных

Давным-давно в одной небольшой компании сотрудники хранили данные о клиентах в Excel. И все работало: сведения были всегда под рукой, а обновлять их не составляло труда. Но время шло, продажи росли, все больше было клиентов и товаров, все больше данных. Таблицы стали огромными, так что теперь на поиск и обновление уходило гораздо больше времени. К тому же местами данные повторялись, местами противоречили друг другу.



И вот однажды в компанию позвонил недовольный клиент, с которого дважды взяли плату за товар. Открыв Excel, менеджер попытался решить проблему, но с ужасом понял, что данные о покупке повреждены и ничего с этим сделать уже нельзя. Вот только клиент такой был не один: вскоре история повторилась, а потом и еще много раз.

Чем чаще такое происходило, тем лучше в компании понимали, что электронные таблицы — не выход: необходима база данных, способная справиться с большим объемом информации, а также обеспечить целостность этой информации.

Если объемы данных невелики, а их структура проста, электронные таблицы вполне подойдут для работы. Действительно, при таких условиях не нужно заботиться о том, как обеспечить согласованность, целостность и безопасность информации, масштабируемость ее структуры, как облегчить ежедневный анализ данных. С увеличением объема и усложнением организации рано или поздно придется задуматься о базе данных.

Реляционные базы данных уже давно являются стандартной технологией хранения и доступа к данным в тех случаях, когда вопросы масштабируемости, согласованности и целостности критически важны. Искусственный интеллект и машинное обучение, бурное развитие которых мы наблюдаем в последние годы, способствуют сохранению и даже росту популярности реляционных баз данных.

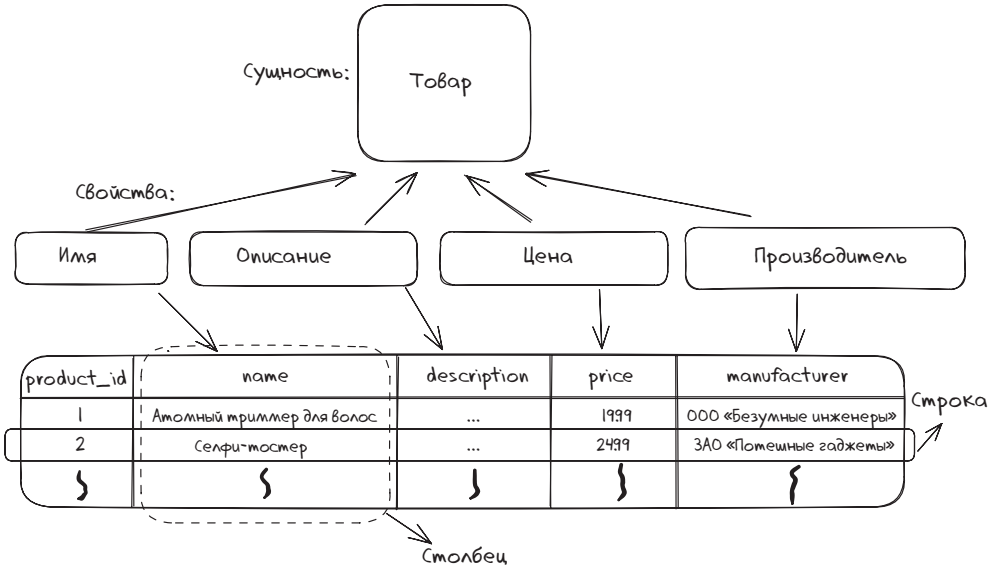
В этом разделе мы познакомимся с основными понятиями из области реляционных баз данных, в том числе таблицами, сущностями и СУБД.

## Таблицы, сущности и первичные ключи

*Реляционная база данных* — это набор таблиц, в которых хранится информация. Эти таблицы похожи на то, что мы видим в Excel, с которым вы наверняка знакомы, — они состоят из строк и столбцов. Таблица может описывать сущность или связь между сущностями: каждая строка таблицы содержит отдельную запись о сущности, а каждый столбец — одно из свойств сущности.

Что такое *сущность* (entity)? Это объект или понятие, которые описываются набором свойств. Допустим, мы обслуживаем интернет-магазин Sci-Fi Collective («Фантастическая лавка»), продающий научно-фантастические товары (к примеру, машину времени, которая вернет вас на пять минут назад, если вы вдруг забудете дома ключи). Каждый такой товар — это сущность, которую можно описать четырьмя свойствами: name (имя),

description (описание), price (цена) и manufacturer (производитель). Если представить список товаров в виде таблицы, каждое из свойств станет столбцом, а запись о каждом товаре — строкой.




Как можно заметить, в этой таблице не четыре, а пять столбцов. Первый из них — `product_id` — это *первичный ключ* (primary key), который нужен для идентификации строк («чтобы всех отыскать, воедино созвать»[1]). В каждой таблице может быть только один такой столбец, а все значения в нем уникальны. Мы будем подробно обсуждать первичные ключи в главе 4.

Нередко в одной таблице Excel хранятся данные о нескольких сущностях. Вы спросите, можно ли сделать так в базе данных? К примеру, объединить информацию о товарах и клиентах «Фантастической лавки», как на рисунке ниже, где добавлено имя клиента (customer_name), ID клиента (customer_id), его электронный адрес (customer_email) и количество (quantity).

[1] Цитата из «Властелина колец» по переводу Н. Григорьевой и В. Грушевского — *Примеч. пер.*

| product_id | name | price | manufacturer | customer_id | customer_name | customer_email | quantity |
|---|---|---|---|---|---|---|---|
| 1 | Атомный триммер … | 19.99 | ООО «Безумные инженеры» | a1 | Боб | bob@gmail.com | 5 |
| 2 | Селфи-мостер | 24.99 | ЗАО «Потешные гаджеты» | b2 | Дэйв | dave@outlook.com | 15 |
| 3 | Кошачий … | 29.99 | АО «Абсурдные аксессуары» | a1 | Боб | bob@gmail.com | 2 |
| … | … | … | … | … | … | … | … |
| 9 | Генератор … | 9.99 | ООО «Всякая всячина» | j8 | Джон | john@123.net | 1 |
| 10 | Нейтрализатор | 33.55 | ООО «Всякая всячина» | p9 | Кейт | katy@123.net | 2 |

Эта таблица — типичный пример плохого дизайна. Помимо очевидной избыточности данных, такая структура чревата неожиданными проблемами. Допустим, какой-то клиент совершил лишь одну покупку и данные о нем есть только в одной строке таблицы. Тогда, удаляя купленный им товар, мы потеряем и сведения о клиенте. Эта проблема известна как *аномалия удаления* (delete anomaly). Еще одна возможная проблема — *аномалия добавления* (insertion anomaly) — будет возникать при добавлении новых строк с товарами. До тех пор, пока товар никто не купил, нам нечем заполнить столбцы, описывающие клиента, однако мы обязаны это сделать, ведь такова структура таблицы. Как результат, мы не сможем внести товар в базу.

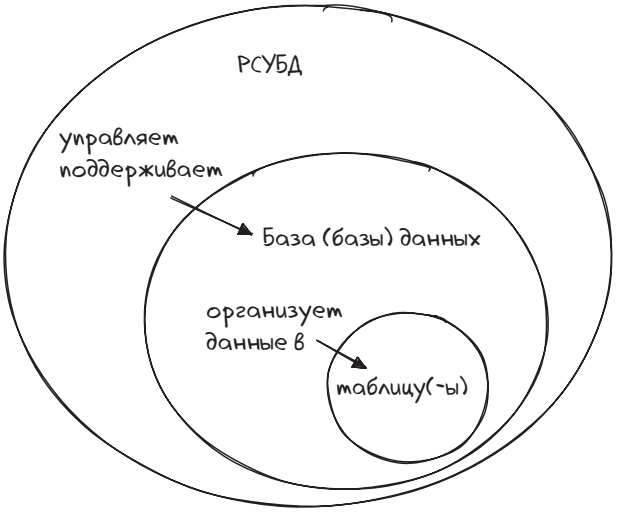
Итак, неудачные решения, принятые при проектировании, могут привести к проблемам, которые неизбежно снизят качество приложения. И чтобы их избежать, нужно освоить базовые принципы и лучшие практики проектирования баз данных.

## Реляционные системы управления базами данных и SQL

Физически управление реляционными базами данных и работу с хранящейся в них информацией осуществляют *реляционные системы управления базами данных* (РСУБД). Впервые, еще в 70-х годах, такую систему создал Эдгар Кодд (Edgar Codd) из IBM.

РСУБД — это специальное ПО, обеспечивающее физическое хранение данных и управление ими. В его состав также входят средства для создания и обновления таблиц, а также для выборки данных. Вы, вероятно, слышали о таких РСУБД, как MySQL, MariaDB, PostgreSQL

и SQLite. Есть и другие. Одна из них непременно потребуется вам, чтобы работать с базой данных, которую вы создадите.




Одно из наиболее примечательных средств, которое поддерживают почти все РСУБД, — SQL, язык программирования, позволяющий создавать таблицы, получать и изменять хранящиеся в них данные. Несмотря на то что реализации SQL в разных РСУБД могут немного отличаться друг от друга, со временем сложился некий общий стандарт этого языка. Поэтому можно сказать, что РСУБД довольно последовательны в отношении SQL, а небольшие различия не имеют особого значения для нашей книги, поскольку речь в ней идет в основном о проектировании.

При разработке баз данных язык SQL можно вообще не использовать: в некоторых РСУБД имеются графические инструменты, которые автоматически генерируют SQL-скрипты, необходимые для создания задуманных баз данных и таблиц. Однако знание этого языка позволит вам лучше понять строение баз данных, особенно в том, что касается параметров структуры и дизайна, в том числе целостности данных, оптимизации и масштабирования. Кроме того, поскольку SQL — стандартизованный язык, который поддерживается большинством РСУБД, знакомство с ним позволит вам меньше зависеть от графических инструментов, а значит, расширит круг доступных вам РСУБД. Поэтому в первых двух главах книги мы и рассмотрим основы этого языка.

# Ваш первый SQL-запрос

В этом разделе мы начнем изучать SQL и выполним первый запрос. Для этого мы воспользуемся примером из прошлого раздела — таблицей из базы данных магазина Sci-Fi Collective («Фантастическая лавка»). В этой базе много таблиц, но сейчас на интересует одна: `product` (Товары), которая выглядит так:

| product_id | name | description | price | manufacturer |
|---|---|---|---|---|
| 1 | Атомный триммер для волос | ... | 19.99 | ООО «Безумные инженеры» |
| 2 | Селфи-мостер | ... | 24.99 | ЗАО «Потешные гаджеты» |
| 3 | Кошачий наполнитель с ароматом кофе | ... | 29.99 | АО «Абсурдные аксессуары» |
| ... | ... | | ... | ... |
| 9 | Генератор нулевой вероятности | ... | 9.99 | ООО «Всякая всячина» |
| 10 | Нейтрализатор | ... | 33.55 | ООО «Всякая всячина» |

Прежде всего, необходимо создать саму базу данных и эту таблицу. Для этого можно использовать готовый SQL-скрипт, который находится в папке `chapter_01` репозитория GitHub к этой книге (https://github.com/Neo-Hao/grokking-relational-database-design). Выполнить этот скрипт в выбранной вами СУБД[1] помогут инструкции из файла `README.md`, который находится в той же папке. Проще всего воспользоваться редактором SQLite Online:

1. Клонируйте или загрузите репозиторий (https://github.com/Neo-Hao/grokking-relational-database-design).
2. Откройте редактор SQLite Online (https://sqliteonline.com).
3. Выберите нужную СУБД на левой панели, а затем нажмите Click to connect (Нажмите, чтобы подключиться).
4. Нажмите Import (Импортировать) и загрузите скрипт для выбранной СУБД (например, для MariaDB используйте файл `mysql_db.sql`).
5. Нажмите Ok.

[1] Здесь и далее под СУБД всегда будет пониматься РСУБД. — *Примеч. ред.*

Теперь мы готовы обратиться с запросом к таблице `product`. Введите следующий код (все три строки) в окне редактора SQLite Online и нажмите Run (Выполнить).

```
SELECT name
FROM product
WHERE price > 20;
```

Что делает этот запрос? Достаточно посмотреть на условие `price > 20` (цена больше 20). Как результат мы получим список товаров из таблицы `product`, стоимость которых превышает `20` (см. ниже). Мы знаем, что в таблице десять позиций, но пять из них (к примеру, атомный триммер для волос) не соответствуют этому условию.

| name |
| --- |
| Селфи-мостер |
| Кошачий наполнитель с ароматом кофе |
| Надувной чемодан |
| Световые мечи |
| Нейтрализатор |

Как можно заметить, язык SQL во многом похож на обычный английский, что необычно. Все дело в том, что большинство языков программирования *императивны*: когда вы пишете код на Java или Python, вы указываете компьютеру, *что* нужно сделать и *как*. Особенность же SQL в том, что он *декларативен*: вы говорите лишь то, *что* следует делать, а всю остальную работу берет на себя СУБД. Мы, люди, намного чаще высказываем пожелания, чем предлагаем способы их осуществления, а потому английский и SQL действительно схожи друг с другом.

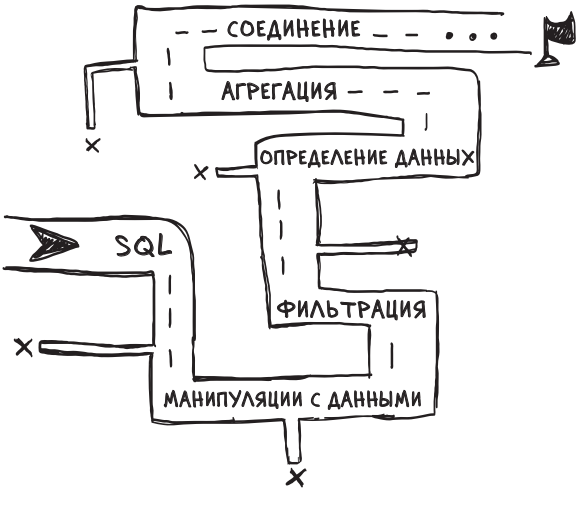
А если точнее, то SQL — это лаконичный английский. Впрочем, нам и не нужен огромный выбор слов и конструкций, чтобы сформулировать *запрос* (query). Для этого достаточно нескольких *выражений* (clause), также называемых *инструкциями* или *операторами* (statement), и небольшого набора правил. Для первого запроса нам потребуется три оператора:

- `SELECT` («выбрать») — указывает, какие столбцы нужно взять из таблицы. В данном случае нас интересуют названия товаров, то есть столбец `name` (имя), а потому мы написали `SELECT name`.
- `FROM` («из») — указывает одну или несколько таблиц, откуда требуется получить данные. Сейчас мы работаем только с одной таблицей: `product`, поэтому написали `FROM product`.
- `WHERE` («где») — задает параметры выбираемых данных. Поскольку нам нужны товары, которые стоят больше 20, следует написать `WHERE price > 20`.

В конце SQL-запроса необходимо поставить точку с запятой (`;`), которая говорит СУБД о том, что текст запроса завершен, а все, что последует дальше, к нему не относится.

# Основы SQL-запросов

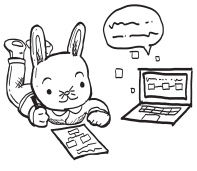
Авторы книги считают, что достаточно изучить лишь самые важные операторы, а все остальные — только по мере необходимости. Операторов в SQL много, однако не все они одинаково важны. Поэтому достаточно нескольких самых распространенных, чтобы усвоить основы и понять, куда двигаться дальше.



Поэтому вместо того чтобы обсуждать все операторы SQL по очереди, мы рассмотрим лишь самые нужные нам для работы, а в этом разделе займемся теми, что составляют запросы к одной таблице.

## Фильтрация

*Фильтрация* (filtering) — одна из самых распространенных задач при выборке данных. Если из общего массива информации необходим лишь определенный набор, подходящий под заданные критерии, нужно отфильтровать данные при помощи условного оператора `WHERE`.



Еще составляя свой первый запрос в этой книге, мы говорили о том, что `WHERE` задает параметры фильтрации данных. К примеру, следующая инструкция позволяет получить из таблицы `product` названия и описания товаров, цена которых не больше `30`:

```
SELECT name, description
FROM product
WHERE price < 30;
```

Когда нас интересует несколько столбцов, мы просто перечисляем их в операторе `SELECT`, используя запятую как разделитель.

А что, если нужно найти все товары конкретного производителя, к примеру ООО «Безумные инженеры» (Mad Inventors Inc)? Используем следующий запрос:

```
SELECT name
FROM product
WHERE manufacturer = 'Mad Inventors Inc';
```

Ответ на него будет таким[1]:

| name |
| --- |
| Атомный триммер для волос |
| Мозговой зонд |
| Световые мечи |

[1] Если вы будете работать с авторскими примерами, взятыми на GitHub, то получите результат на английском языке (см. раздел «От издательства» в начале книги). — *Примеч. ред.*

В этом запросе для проверки используется одиночный знак равенства (=). Также необходимо обратить внимание на одинарные кавычки (' '), при помощи которых выделяются строковые данные. Вы спросите: в SQL есть разные типы данных? Конечно. И все их можно разделить на шесть категорий:

- числовые данные (например, INT);
- строковые данные (например, TEXT);
- временны́е данные (например, DATE);
- строковые данные в формате Unicode (например, VARCHAR);
- двоичные данные (например, BINARY);
- прочие данные (например, XML).

Столбец manufacturer (Производитель) таблицы product относится к строковому типу. В отличие от него столбец price (Цена) является числовым.

Освоив фильтрацию числовых и строковых данных, поговорим о том, как объединять параметры в одном фильтре при помощи логических операторов. Чаще всего используются AND (И) и OR (ИЛИ), назначение которых несложно понять из перевода. К примеру, можно объединить два ранее упомянутых параметра при помощи оператора AND следующим образом:

```
SELECT *
FROM product
WHERE price < 30 AND manufacturer = 'Mad Inventors Inc.';
```

В результате получим:

| product_id | name | description | price | manufacturer |
|---|---|---|---|---|
| 1 | Атомный триммер для волос | ... | 1999 | ООО «Безумные инженеры» |
| 6 | Мозговой зонд | ... | 1999 | ООО «Безумные инженеры» |
| 7 | Световые мечи | ... | 25.00 | ООО «Безумные инженеры» |

В отличие от прошлых запросов, в этом после SELECT мы указали не список столбцов из таблицы product, а звездочку (*), что означает, что нам нужны все столбцы. Благодаря фильтрации по двум параметрам мы получили данные о товарах, изготовленных ООО «Безумные инженеры» и сто́ящих не более 30.