УДК 004.655
ББК 32.973.26-018.2
М79

Моргунов, Е. П.

М79 PostgreSQL. Основы языка SQL: учеб. пособие / Е. П. Моргунов; под ред. Е. В. Рогова, П. В. Лузанова. — СПб.: БХВ-Петербург, 2018. — 336 с.: ил.

ISBN 978-5-9775-4022-3

Учебно-практическое пособие охватывает первую, базовую, часть учебного курса по языку SQL, созданного при участии российской компании Postgres Professional. Учебный материал излагается в расчете на использование системы управления базами данных PostgreSQL. Рассмотрено создание рабочей среды, описаны языки определения данных и основные операции выборки и изменения данных. Показаны примеры использования транзакций, уделено внимание методам оптимизации запросов. Материал сопровождается многочисленными практическими примерами. Пособие может использоваться как для самостоятельного обучения, так и при проведении занятий под руководством преподавателя.

Для программистов и студентов

УДК 004.655
ББК 32.973.26-018.2

Рецензент: *Б. А. Новиков*, д-р. физ.-мат. наук, профессор СПбГУ

ISBN 978-5-9775-4022-3
ISBN 978-5-6041193-2-7

Оглавление

Предисловие автора

С системой управления базами данных PostgreSQL мне довелось познакомиться в 1998 году, это была еще версия 6.2. Важную роль в моей встрече с PostgreSQL сыграл Вадим Михеев, который в те годы был одним из ключевых разработчиков этой СУБД. После перехода в 2000 году на преподавательскую работу в вуз мне потребовалась свободно распространяемая СУБД для использования в учебном процессе. На эту роль идеально подходила СУБД PostgreSQL.

Книга, которую вы держите в руках, появилась по инициативе и при поддержке компании Postgres Professional — российского поставщика СУБД PostgreSQL. Летом 2016 года я встретился с директором компании Олегом Бартуновым. Он высказал идею о том, что для распространения СУБД PostgreSQL необходим учебник по языку SQL. PostgreSQL является самой продвинутой свободно распространяемой СУБД с очень хорошей поддержкой стандарта SQL. Тем не менее существуют и специфические черты, присущие именно PostgreSQL, поэтому такой учебник должен был отражать эту специфику. Поскольку я уже много лет использовал PostgreSQL в преподавании в вузе, то Олег Бартунов предложил мне заняться написанием такого учебника. Это предложение показалось мне очень интересным.

Мы обсудили план будущего учебника с Павлом Лузановым и Егором Роговым, которые занимаются образовательными программами компании. Они согласились выступить в трудной и ответственной роли редакторов книги. Координировали работу над учебником и помогали на разных стадиях этого трудного процесса Олег Бартунов и заместитель директора компании Иван Панченко. Оригинальную обложку книги сделал Андрей Климковский, а Егор Рогов подготовил предметный указатель, благодаря которому книгой будет удобно пользоваться. Вообще, без поддержки компании Postgres Professional учебник в таком виде не смог бы состояться.

Данный учебник рекомендуется использовать в комплексе с учебником по основам технологий баз данных, который написан профессором Санкт-Петербургского государственного университета Борисом Асеновичем Новиковым также при поддержке компании Postgres Professional. Важным является то, что в обоих учебниках для иллюстрации приемов применения языка SQL используется одна и та же учебная база данных «Авиаперевозки», разработанная Павлом Лузановым и Егором Роговым. Она содержит данные, по своим свойствам близкие к реальным. Это позволяет в качестве упражнений моделировать ситуации, которые могли бы иметь место в реальной работе специалиста по базам данных.

Знанием только языка SQL квалификация специалиста по базам данных не исчерпывается, поэтому на сайте компании Postgres Professional представлен целый ряд «фирменных» учебных курсов по PostgreSQL. Ссылка на них приведена в конце книги в разделе «Рекомендуемые источники».

В планах автора и компании Postgres Professional — выпуск второй части учебника. Она будет посвящена более сложным вопросам использования языка SQL в среде СУБД PostgreSQL. Предполагается рассмотреть такие темы, как хранимые функции и процедуры, язык PL/pgSQL, триггеры, полнотекстовый поиск, расширяемость PostgreSQL.

Е. П. Моргунов

Введение

В настоящее время термин «база данных» известен многим людям, даже далеким от профессиональной разработки компьютерных программ. Базы данных стали очень широко распространенной технологией, что потребовало, в свою очередь, большего числа специалистов, способных проектировать их и обслуживать. В ходе эволюции теории и практики баз данных стандартом де-факто стала реляционная модель данных, а в рамках этой модели сформировался и специализированный язык программирования, позволяющий выполнять все необходимые операции с данными — Structured Query Language (SQL). Таким образом, важным компонентом квалификации специалиста в области баз данных является владение языком SQL.

В настоящем учебном пособии излагаются основы языка SQL — это базовый курс. Причем язык рассматривается применительно к конкретной системе управления базами данных (СУБД) — PostgreSQL. Реализация языка SQL в каждой СУБД соответствует стандарту в той или иной степени, но кроме стандартизированных функций и возможностей, каждая СУБД предлагает и свои дополнительные расширения языка. PostgreSQL обеспечивает очень хорошую поддержку стандарта языка SQL и также предоставляет интересные и практически полезные дополнительные возможности. Одним из главных достоинств PostgreSQL является расширяемость. Это означает, например, что пользователь (конечно, являющийся специалистом в области баз данных) может разработать свои собственные типы данных. Эти типы данных будут обладать всеми свойствами встроенных типов данных и могут быть введены в работу без останова сервера. Кроме того, PostgreSQL является свободно распространяемым продуктом с открытым исходным кодом, который доступен на большом числе платформ.

В пособии рассматриваются не только все основные команды языка SQL, но также и другие вопросы, такие как индексы и транзакции.

Книга написана таким образом, чтобы ее можно было использовать как под руководством преподавателя, так и самостоятельно. Предполагается, что студенты имеют доступ к уже установленной СУБД, поэтому процедура установки PostgreSQL детально не рассматривается, а лишь даются указания о том, где найти инструкции по установке.

Это пособие предназначено для получения практических навыков использования языка SQL. Учебный материал подается таким образом, что многие важные знания

читатель должен получить в результате выполнения заданий, находящихся в конце каждой главы. В основном тексте глав эти знания могут быть не представлены. Предполагается, что значительная часть заданий будет выполняться читателем самостоятельно с помощью документации на СУБД PostgreSQL, но зачастую даются и указания к их выполнению. Задания, приведенные в пособии, различаются по уровню сложности. Самые сложные из них, а также те, которые требуют много времени для выполнения, отмечены звездочкой.

Задания можно выполнять по мере изучения учебного материала конкретной главы. Однако некоторые из них имеют комплексный характер, поэтому для их выполнения необходимо изучить всю главу или, как минимум, несколько ее разделов.

Хотя пособие имеет практическую направленность и не является теоретическим курсом, все же в первой главе кратко, на элементарном уровне излагаются основные понятия теории баз данных и реляционной модели. Это сделано для того, чтобы студенты могли приступить к практическому освоению языка SQL без задержки, с первых дней учебного семестра, еще до того момента, когда эти понятия будут основательно рассмотрены в лекционном курсе.

На факультетах информационных технологий в российских вузах базы данных традиционно изучаются на втором или третьем курсе. Причем этой дисциплине, как правило, отводится один семестр. Однако количество академических учебных часов может различаться. Если на практические занятия по этой дисциплине учебный план отводит 36 часов, тогда мы рекомендуем следующее распределение времени на изучение материала пособия.

Глава 1. Введение в базы данных и SQL	1 час
Глава 2. Создание рабочей среды	1 час
Глава 3. Основные операции с таблицами	4 часа
Глава 4. Типы данных СУБД PostgreSQL	4 часа
Глава 5. Основы языка определения данных	4 часа
Глава 6. Запросы	8 часов
Глава 7. Изменение данных	4 часа
Глава 8. Индексы	2 часа
Глава 9. Транзакции	4 часа
Глава 10. Повышение производительности	4 часа

Главы 1 и 2 могут быть изучены за одно двухчасовое занятие, поскольку система PostgreSQL уже должна быть установлена в учебной аудитории заранее.

Глава 3 представляет собой краткий обзор основных возможностей языка SQL, после ее изучения студенты должны представлять себе простые способы использования

всех основных команд языка. Эта глава не очень сложная, но объемная, поэтому на ее изучение отводится четыре часа.

Глава 4 посвящена рассмотрению основных типов данных, используемых в СУБД PostgreSQL. Это большая глава, однако в ней значительную часть составляют задания и упражнения. Предполагается, что студенты за четыре часа должны усвоить только основные приемы использования типов данных. А для того, чтобы знания закрепились, рекомендуется обращаться к материалу этой главы (в том числе и к упражнениям) в процессе изучения остальных глав пособия при необходимости уточнения тех или иных особенностей применения конкретных типов. Распределить время, выделенное на изучение этой главы, мы рекомендуем следующим образом: два часа на первые четыре раздела — числовые и строковые типы, типы «дата/время» и логический тип, еще два часа — на массивы и тип `json`/`jsonb`.

Чтобы выполнять запросы к базе данных, необходимо хорошо понимать ее структуру, взаимосвязи таблиц. Поэтому глава 5, в которой рассматриваются основы языка определения данных, очень важна с точки зрения детального изучения таблиц базы данных «Авиаперевозки» и подготовки к освоению главы 6. Поскольку материал главы основан на том, что база данных уже развернута на компьютере студента, то вводить команды для создания таблиц не требуется. Это позволяет сократить время, затрачиваемое на изучение главы. В пособии принят подход, при котором сначала рассматриваются команды определения данных, а затем — команды манипулирования данными. Поэтому глава 5 «Основы языка определения данных» предшествует главе 6 «Запросы». Однако избранный подход реализуется не слишком жестко: в обзорной главе 3 рассматриваются основные команды, в том числе и несложные запросы. А запросы — это уже язык манипулирования данными. На изучение главы 5 отведено четыре часа. В течение первого двухчасового занятия нужно изучить два первых раздела, в которых освещаются такие вопросы, как ограничения целостности и создание и удаление таблиц. Второе двухчасовое занятие нужно посвятить изучению трех оставшихся разделов. В них говорится о способах модификации таблиц, а также о представлениях (views) и схемах базы данных.

Глава 6 является центральной главой пособия, поэтому на ее изучение отводится восемь часов, т. е. больше, чем на изучение других глав. Она состоит из четырех разделов. Первый из них посвящен разнообразным дополнительным возможностям команды `SELECT`. Речь идет, в частности, о таких вещах, как предложения `LIMIT` и `OFFSET`, оператор `LIKE` и регулярные выражения в условиях предложения `WHERE` и о других возможностях. Тем не менее материал этого раздела несложный, для его изучения достаточно выделить один час. Во втором разделе рассказывается о способах соединения таблиц. Это более сложная тема, поэтому для ее изучения необходимо выделить два часа. Третий раздел посвящен агрегированию и группировке. В нем

рассматривается и такая важная и интересная тема, как оконные функции. Данный раздел также требует двухчасового занятия. Самый сложный раздел этой главы — четвертый. Он посвящен подзапросам. В нем, в частности, освещается такая важная и интересная тема, как общие табличные выражения (Common Table Expressions — CTE). Для изучения материала данного раздела необходимо выделить три часа.

В главе 7 собраны все команды, предназначенные для изменения данных: вставка строк, их обновление и удаление. Поскольку в предшествующих главах эти команды уже применялись для решения простых задач, то в данной главе рассматриваются более сложные способы их использования. В ней много упражнений, они составляют половину ее объема. Рекомендуется уделить два часа изучению способов вставки строк в таблицы, а еще два часа — рассмотрению операций обновления и удаления строк.

Глава 8 посвящена индексам, она небольшая, поэтому с ней можно ознакомиться за одно двухчасовое занятие. Поскольку индексы тесно связаны с вопросами производительности, т. е. скорости выполнения запросов, то было бы целесообразно после изучения заключительной главы вернуться к главе 8 и посмотреть на представленные в ней команды и запросы, уже зная о команде `EXPLAIN`.

Транзакциям посвящена глава 9. Механизмы их выполнения имеют много тонкостей, поэтому при изучении этой главы необходимо экспериментировать и стараться объяснить полученные результаты.

В заключительной главе 10 рассматриваются вопросы повышения производительности. Эта глава может показаться слишком абстрактной и сложной для начального курса языка SQL, тем не менее она очень важна. Студенты должны научиться читать планы выполнения запросов и понимать назначение каждой операции, представленной в плане. А овладение искусством оптимизации запросов потребует много времени и опыта, оно придет не сразу.

В том случае, когда на практические занятия по дисциплине «Базы данных» в учебном плане отводится 54 часа, можно изменить предлагаемое распределение учебных часов. В частности, в главе 4 можно больше времени посвятить типам данных `json`/`jsonb` и массивам. В главе 6 можно более детально рассмотреть оконные функции и общие табличные выражения. При изучении главы 9, посвященной транзакциям, было бы целесообразно разработать несложное приложение, в котором использовались бы транзакции, и провести эксперименты с этим приложением, выполняя параллельно несколько сеансов и изменяя при этом уровни изоляции транзакций. В рамках главы 10 имеет смысл вернуться к командам и запросам главы 8 и изучить

планы их выполнения с помощью команды EXPLAIN. За счет дополнительного времени можно рассмотреть все задания и упражнения повышенной сложности (помеченные звездочкой).

Таким образом, распределение времени может быть таким:

Глава 1. Введение в базы данных и SQL	1 час
Глава 2. Создание рабочей среды	1 час
Глава 3. Основные операции с таблицами	4 часа
Глава 4. Типы данных СУБД PostgreSQL	6 часов
Глава 5. Основы языка определения данных	6 часов
Глава 6. Запросы	12 часов
Глава 7. Изменение данных	6 часов
Глава 8. Индексы	4 часа
Глава 9. Транзакции	8 часов
Глава 10. Повышение производительности	6 часов

В пособии используются различные виды шрифтов для выделения фрагментов текста в зависимости от их назначения. Команды, вводимые пользователем как в среде операционной системы, так и в среде утилиты psql, выделяются полужирным моноширинным шрифтом. Например:

```
psql -d demo -U postgres
```

или

```
SELECT avg( total_amount ) FROM bookings;
```

Результаты работы команд операционной системы и SQL-команд, выполняемых в среде утилиты psql, напечатаны моноширинным шрифтом. Например, в ответ на команду

```
EXPLAIN SELECT * FROM aircrafts;
```

на экран будет выведено следующее:

```
                       QUERY PLAN
-------------------------------------------------------------
Seq Scan on aircrafts (cost=0.00..1.09 rows=9 width=52)
(1 строка)
```

Мы надеемся, что изучение материала, изложенного в учебном пособии, будет способствовать повышению уровня вашей квалификации и расширению профессионального кругозора.

Глава 1

Введение в базы данных и SQL

Эта глава — вводная. В ней мы расскажем об основах баз данных, о том, что такое реляционная модель и зачем нужен язык SQL. Очень важной темой этой главы станет описание предметной области, на основе которой будет спроектирована учебная база данных, которая и будет служить в качестве площадки для изучения языка SQL. Это пособие предназначено в первую очередь для практического освоения языка SQL, а не для изучения теории баз данных, поэтому для изучения теории необходимо обращаться к авторитетным источникам, список которых приведен в конце учебного пособия.

1.1. Что такое базы данных и зачем они нужны

Технологии баз данных существовали не всегда. Однако и до их внедрения в практику люди также собирали и обрабатывали данные. Одним из способов хранения данных были так называемые плоские файлы (flat files), которые имели очень простую структуру: данные хранились в виде записей, разделенных на поля фиксированной длины. В реальной жизни между элементами данных зачастую возникают сложные связи, которые необходимо перенести и в электронную базу данных. При использовании плоских файлов эти связи организовать сложно, а еще сложнее поддерживать их при изменениях и удалениях отдельных элементов данных.

Одним из основных понятий в теории баз данных является **модель данных**. Можно сказать, что она характеризует способ организации данных и основные методы доступа к ним. Сначала были предложены иерархическая и сетевая модели данных. Однако в ходе эволюции теорий и идей была разработана реляционная модель данных, которая сейчас и является доминирующей. Поэтому в настоящее время преобладают базы данных реляционного типа. Их характерной чертой является тот факт, что данные воспринимаются пользователем как таблицы. В распоряжении пользователя имеются операторы для выборки данных из таблиц, а также для вставки новых данных, обновления и удаления имеющихся данных.

Одним из достоинств реляционной базы данных является ее способность поддерживать связи между элементами данных, избавляя программиста от необходимости заниматься этой рутинной и очень трудоемкой работой. В те времена, когда технологии

реляционных баз данных еще не получили широкого распространения, программистам приходилось на процедурных языках вручную реализовывать такие операции, которые сейчас называются каскадным обновлением внешних ключей или каскадным удалением записей из подчиненных таблиц (файлов). Здесь слово «вручную» означает, что для выполнения этих операций приходилось писать код, состоящий из элементарных команд, позволяющий добраться до каждой обновляемой или удаляемой записи. Тот подход к работе с базами данных назывался навигационным — программист указывал программе конкретный алгоритм поиска записей. Приведем в качестве примера простую ситуацию: в базе данных, построенной на основе файлов, хранится информация о студентах и их экзаменационных оценках, причем личные данные студентов хранятся в одном файле, назовем его условно «Студенты», а экзаменационные оценки — в другом файле, который условно назовем «Успеваемость». Если требуется удалить информацию о конкретном студенте и его экзаменационных оценках, то придется не только выполнить операцию удаления конкретной записи из файла «Студенты», но дополнительно организовать цикл для поиска и удаления тех записей из файла «Успеваемость», у которых ключевое поле имеет то же значение, что и поле в удаляемой записи из файла «Студенты».

Работая с реляционными базами данных, программист избавлен от программирования на «атомарном» уровне, потому что современные языки для «общения» с этими базами данных являются декларативными. Это означает, что для получения результата достаточно лишь указать, *что* нужно получить, но не требуется предписывать способ получения результата, т. е. *как* его получить.

Система баз данных — это компьютеризированная система, предназначенная для хранения, переработки и выдачи информации по запросу пользователей. Такая система включает в себя программное и аппаратное обеспечение, сами данные, а также пользователей.

Современные системы баз данных являются, как правило, многопользовательскими. В таких системах одновременный доступ к базе данных могут получить сразу несколько пользователей.

Основным программным обеспечением является система управления базами данных. По-английски она называется database management system (DBMS). Кроме СУБД в систему баз данных могут входить утилиты, средства для разработки приложений (программ), средства проектирования базы данных, генераторы отчетов и др.

Пользователи систем с базами данных подразделяются на ряд категорий. Первая категория — это прикладные программисты. Вторая категория — это конечные пользователи, ради которых и выполняется вся работа. Они могут получить доступ к базе

данных, используя прикладные программы или универсальные приложения, которые входят в программное обеспечение самой СУБД. В большинстве СУБД есть так называемый **процессор языка запросов**, который позволяет пользователю вводить команды языка высокого уровня (например, языка SQL). Третья категория пользователей — это администраторы базы данных. В их обязанности входят: создание базы данных, выбор оптимальных режимов доступа к ней, разграничение полномочий различных пользователей на доступ к той или иной информации в базе данных, выполнение резервного копирования базы данных и т. д.

Систему баз данных можно разделить на два главных компонента: сервер и набор клиентов (или внешних интерфейсов). Сервер — это и есть СУБД. Клиентами являются различные приложения, написанные прикладными программистами, или встроенные приложения, поставляемые вместе с СУБД. Один сервер может обслуживать много клиентов.

Современные СУБД включают в себя словарь данных. Это часть базы данных, которая описывает сами данные, хранящиеся в ней. Словарь данных помогает СУБД выполнять свои функции.

1.2. Основные понятия реляционной модели

В каждой технологической сфере есть своя терминология. Существуют базовые термины, на которых основываются все дальнейшие рассуждения. Такие термины присутствуют и в сфере баз данных. Сейчас мы кратко о них поговорим.

В эпоху, предшествующую рождению реляционной теории, базы данных традиционно рассматривались как набор **файлов**, состоящих из **записей**, а записи, в свою очередь, подразделялись на отдельные **поля**. Поле являлось элементарной единицей данных.

В реляционных базах данных пользователь воспринимает данные в виде таблиц. Поэтому термину «файл» соответствует термин **«таблица»**, вместо термина «запись» используется термин **«строка»**, а вместо термина **«поле»** — термин **«столбец»** (или **«колонка»**). Таким образом, таблицы состоят из строк и столбцов, на пересечении которых должны находиться «атомарные» значения, которые нельзя разбить на более мелкие элементы без потери смысла.

В формальной теории реляционных баз данных эти таблицы называют **отношениями (relations)** — поэтому и базы данных называются реляционными. Отношение —

это математический термин. При определении свойств таких отношений используется теория множеств. В терминах данной теории строки таблицы будут называться **кортежами (tuples)**, а колонки — **атрибутами**. Отношение имеет заголовок, который состоит из атрибутов, и тело, состоящее из кортежей. Количество атрибутов называется **степенью отношения**, а количество кортежей — **кардинальным числом**. Кроме теории множеств, одним из оснований реляционной теории является такой раздел математической логики, как исчисление предикатов.

Таким образом, в теории и практике баз данных существует три группы терминов. Иногда термины из разных групп используют в качестве синонимов, например, запись и строка.

Как мы уже сказали выше, в реляционных базах данных пользователь воспринимает данные в виде таблиц.

Рассмотрим простую систему, в которой всего две таблицы. Первая — «Студенты»:

№ зачетной книжки	Ф. И. О.	Серия документа	Номер документа
55500	Иванов Иван Петрович	0402	645327
55800	Климов Андрей Иванович	0402	673211
55865	Новиков Николай Юрьевич	0202	554390

И вторая — «Успеваемость»:

Зачетная книжка	Предмет	Учебный год	Семестр	Оценка
55500	Физика	2016/2017	1	5
55500	Математика	2016/2017	1	4
55800	Физика	2016/2017	1	4
55800	Физика	2016/2017	2	3

При работе с базами данных часто приходится следовать различным **ограничениям**, которые могут быть обусловлены спецификой конкретной предметной области. Упрощая реальную ситуацию, примем следующие ограничения:

– номер зачетной книжки состоит из пяти цифр и не может быть отрицательным (в разных вузах используются различные схемы присваивания номеров зачетным книжкам, эти схемы могут быть гораздо сложнее принятой нами и могут учитывать, например, год поступления студента в вуз);

– серия документа, удостоверяющего личность, является четырехзначным числом, а номер документа, удостоверяющего личность, — шестизначным числом;

– номер семестра может принимать только два значения — 1 (осенний семестр) и 2 (весенний семестр);

– оценка может принимать только три значения — 3 (удовлетворительно), 4 (хорошо) и 5 (отлично): другие оценки выставлять в зачетные книжки не принято.

Для идентификации строк в таблицах и для связи таблиц между собой используются так называемые ключи. **Потенциальный ключ** — это комбинация атрибутов таблицы, позволяющая уникальным образом идентифицировать строки в ней. Ключ может состоять только лишь из одного атрибута таблицы. Например, в таблице «Студенты» таким идентификатором может быть атрибут «Номер зачетной книжки». В качестве потенциального ключа данной таблицы могут также служить два ее атрибута, взятые вместе: «Серия документа, удостоверяющего личность» и «Номер документа, удостоверяющего личность». Ни один из них в отдельности не может использоваться в качестве уникального идентификатора. В таком случае ключ будет составным. При этом важным является то, что потенциальный ключ должен быть *неизбыточным*, т. е. никакое подмножество атрибутов, входящих в него, не должно обладать свойством уникальности. Потенциальный ключ, включающий два упомянутых атрибута, является неизбыточным.

Ключи нужны для адресации на уровне строк (записей). При наличии в таблице более одного потенциального ключа один из них выбирается в качестве так называемого **первичного ключа**, а остальные будут являться **альтернативными ключами**.

Рассмотрим таблицы «Студенты» и «Успеваемость». Предположим, что в таблице «Студенты» нет строки с номером зачетной книжки 55900, тогда включать строку с таким номером зачетной книжки в таблицу «Успеваемость» не имеет смысла. Таким образом, значения столбца «Номер зачетной книжки» в таблице «Успеваемость» должны быть согласованы со значениями такого же столбца в таблице «Студенты». Атрибут «Номер зачетной книжки» в таблице «Успеваемость» является примером того, что называется **внешним ключом**. Таблица, содержащая внешний ключ, называется **ссылающейся** таблицей (referencing table). Таблица, содержащая соответствующий потенциальный ключ, называется **ссылочной (целевой)** таблицей (referenced table). В таких случаях говорят, что внешний ключ ссылается на потенциальный ключ в ссылочной таблице. Внешний ключ может быть составным, т. е. может включать более одного атрибута. Внешний ключ не обязан быть уникальным. Проблема обеспечения того, чтобы база данных не содержала неверных значений внешних ключей, известна как проблема **ссылочной целостности**. Ограничение, согласно которому значения внешних ключей должны соответствовать значениям потенциальных ключей, называется **ограничением ссылочной целостности (ссылочным ограничением)**.

Обеспечением выполнения ограничений ссылочной целостности занимается СУБД, а от разработчика требуется лишь указать атрибуты, служащие в качестве внешних ключей. При проектировании баз данных часто предусматривается, что при удалении строки из ссылочной таблицы соответствующие строки из ссылающейся таблицы должны быть также удалены, а при изменении значения столбца, на который ссылается внешний ключ, должны быть изменены значения внешнего ключа в ссылающейся таблице. Этот подход называется **каскадным удалением (обновлением)**.

Иногда применяются и другие подходы. Например, вместо удаления строк из ссылающейся таблицы в этих строках просто заменяют значения атрибутов, входящих во внешний ключ, так называемыми NULL-значениями. Это специальные значения, означающие «ничто» или отсутствие значения, они не совпадают со значением «нуль» или «пустая строка». NULL-значение применяется в базах данных и в качестве значения по умолчанию, когда пользователь не ввел никакого конкретного значения. Первичные ключи не могут содержать NULL-значений.

Транзакция — одно из важнейших понятий теории баз данных. Она означает набор операций над базой данных, рассматриваемых как единая и неделимая единица работы, выполняемая полностью или не выполняемая вовсе, если произошел какой-то сбой в процессе выполнения транзакции. Таким образом, транзакции являются средством обеспечения согласованности данных. В нашей базе данных транзакцией могут быть, например, две операции: удаление строки из таблицы «Студенты» и удаление связанных по внешнему ключу строк из таблицы «Успеваемость».

1.3. Что такое язык SQL

Язык SQL — это непроцедурный язык, который является стандартным средством работы с данными во всех реляционных СУБД. Операторы (команды), написанные на этом языке, лишь указывают СУБД, какой результат должен быть получен, но не описывают процедуру получения этого результата. СУБД сама определяет способ выполнения команды пользователя. В языке SQL традиционно выделяются группа операторов определения данных (Data Definition Language — DDL), группа операторов манипулирования данными (Data Manipulation Language — DML) и группа операторов, управляющих привилегиями доступа к объектам базы данных (Data Control Language — DCL).

К операторам языка определения данных (DDL) относятся команды для создания, изменения и удаления таблиц, представлений и других объектов базы данных. Детальному рассмотрению этих команд посвящены главы 5 и 8.

К операторам языка манипулирования данными (DML) относятся команды для выборки строк из таблиц, вставки строк в таблицы, обновления и удаления строк. Эти команды подробно рассматриваются в главах 6 и 7.

Операторы DCL в пособии не рассматриваются, т. к. PostgreSQL позволяет на начальном этапе изучения языка SQL обойтись без их использования.

1.4. Описание предметной области и учебной базы данных

Чтобы показать все основные возможности языка SQL, нам потребуется база данных. Эта база данных не должна быть слишком сложной, чтобы ее изучение не потребовало слишком много времени. Но вместе с тем она должна быть достаточно разнообразной, чтобы запросы к ней выглядели бы правдоподобными, почти такими же, как и в реальной работе.

В качестве предметной области выберем пассажирские авиаперевозки. Ее оригинальное описание и описание базы данных «Авиаперевозки» можно найти по адресам https://postgrespro.ru/education/demodb и https://postgrespro.ru/docs/postgrespro/current/demodb-bookings.html. Надеемся, что эта область знакома многим читателям нашего учебного пособия. Конечно, в учебных целях реальная ситуация намеренно упрощена, но все принципиальные вещи сохранены.

Итак, некая российская авиакомпания выполняет пассажирские авиаперевозки. Она обладает своим парком самолетов различных моделей. Каждая модель самолета имеет определенный код, который присваивает Международная ассоциация авиаперевозчиков (IATA). При этом будем считать, что самолеты одной модели имеют одинаковые компоновки салонов, т. е. порядок размещения кресел и нумерацию мест в салонах бизнес-класса и экономического класса. Например, если это модель Sukhoi SuperJet-100, то место 2A относится к бизнес-классу, а место 20D — к экономическому классу. Бизнес-класс и экономический класс — это разновидности так называемого класса обслуживания.

Наша авиакомпания выполняет полеты между аэропортами России. Каждому аэропорту присвоен уникальный трехбуквенный код, при этом используются только заглавные буквы латинского алфавита. Эти коды присваивает не сама авиакомпания, а специальные организации, управляющие пассажирскими авиаперевозками. Зачастую название аэропорта не совпадает с названием того города, которому этот аэропорт принадлежит. Например, в городе Новосибирске аэропорт называется Толмачево, в городе Екатеринбурге — Кольцово, а в Санкт-Петербурге — Пулково. К тому же

некоторые города имеют более одного аэропорта. Сразу в качестве примера вспоминается Москва с ее аэропортами Домодедово, Шереметьево и Внуково. Добавим еще одну важную деталь: каждый аэропорт характеризуется географическими координатами — долготой и широтой, а также часовым поясом.

Формируются маршруты перелетов между городами. Конечно, каждый такой маршрут требует указания не только города, но и аэропорта, поскольку, как мы уже сказали, в городе может быть и более одного аэропорта. В качестве упрощения реальности мы решим, что маршруты не будут иметь промежуточных посадок, т. е. у них будет только аэропорт отправления и аэропорт назначения. Каждый маршрут имеет шестизначный номер, включающий цифры и буквы латинского алфавита.

На основе перечня маршрутов формируется расписание полетов (или рейсов). В расписании указывается плановое время отправления и плановое время прибытия, а также тип самолета, выполняющего этот рейс.

При фактическом выполнении рейса возникает необходимость в учете дополнительных сведений, а именно: фактического времени отправления и фактического времени прибытия, а также статуса рейса. Статус рейса может принимать ряд значений:

- Scheduled (за месяц открывается возможность бронирования);
- On Time (за сутки открывается регистрация);
- Delayed (рейс задержан);
- Departed (вылетел);
- Arrived (прибыл);
- Cancelled (отменен).

Теперь обратимся к пассажирам. Полет начинается с бронирования авиабилета. В настоящее время общепринятой практикой является оформление электронных билетов. Каждый такой билет имеет уникальный номер, состоящий из 13 цифр. В рамках одной процедуры бронирования может быть оформлено несколько билетов, но каждая такая процедура имеет уникальный шестизначный номер (шифр) бронирования, состоящий из заглавных букв латинского алфавита и цифр. Кроме того, для каждой процедуры бронирования записывается дата бронирования и рассчитывается общая стоимость оформленных билетов.

В каждый билет, кроме его тринадцатизначного номера, записывается идентификатор пассажира, а также его имя и фамилия (в латинской транскрипции) и контактные данные. В качестве идентификатора пассажира используется номер документа, удостоверяющего личность. Конечно, пассажир может сменить свой документ, а иной

раз даже фамилию и имя, за время, прошедшее между бронированием билетов в разные дни, поэтому невозможно наверняка сказать, что какие-то конкретные билеты были оформлены на одного и того же пассажира.

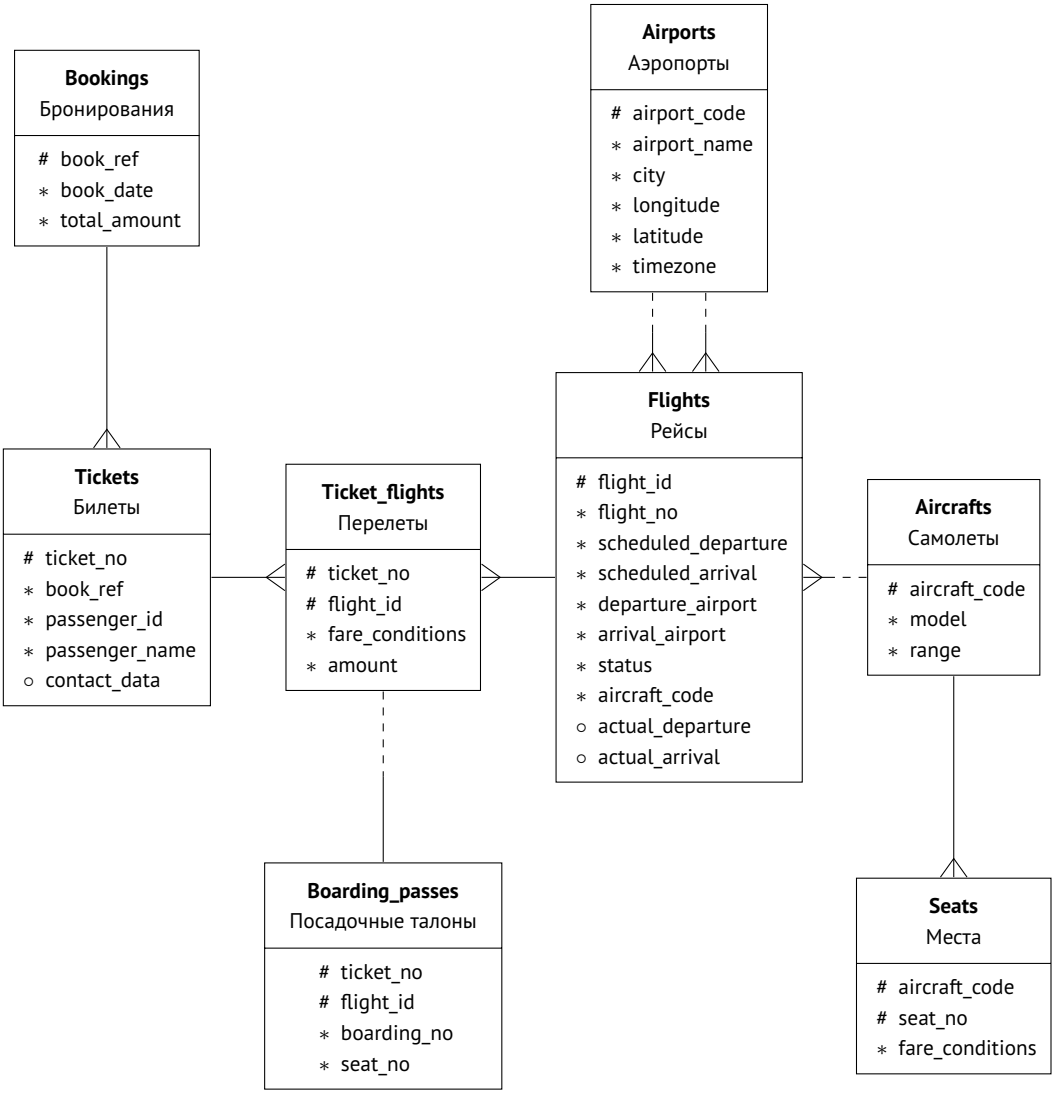
В каждый электронный билет может быть вписано более одного перелета. Специалисты называют эти записи о перелетах сегментами. В качестве примера наличия нескольких сегментов можно привести такой: Красноярск — Москва, Москва — Анапа, Анапа — Москва, Москва — Красноярск. При этом возможно в рамках одного бронирования оформить несколько билетов на различных пассажиров. Для каждого перелета указывается номер рейса, аэропорты отправления и назначения, время вылета и время прибытия, а также стоимость перелета. Кроме того, указывается и так называемый класс обслуживания: экономический, бизнес и др.

Когда пассажир прибывает в аэропорт отправления и проходит регистрацию билета, оформляется так называемый посадочный талон. Этот талон связан с авиабилетом: в талоне указывается такой же номер, который имеет электронный авиабилет данного пассажира. Кроме того, в талоне указывается номер рейса и номер места в самолете. Указывается также и номер посадочного талона — последовательный номер, присваиваемый в процессе регистрации билетов на данный рейс.

Напомним, что каждому креслу в салоне самолета соответствует конкретный класс обслуживания. Данная информация учитывается при регистрации билетов и оформлении посадочных талонов. Если, например, пассажир приобрел билет с экономическим классом обслуживания, то в его посадочном талоне будет указан номер места в салоне экономического класса, но не в салоне бизнес-класса.

Следуя приведенному описанию предметной области, можно спроектировать модельную базу данных, предназначенную для изучения языка SQL. Поскольку наше учебное пособие в первую очередь предназначено для изучения языка SQL, а не для овладения искусством проектирования баз данных, то мы приведем лишь диаграмму схемы данных, на которой показаны сущности, выделенные в предметной области, а также их связи и атрибуты. Конкретные же типы данных, первичные и внешние ключи, а также ограничения, наложенные на атрибуты и таблицы, мы покажем уже в последующих главах в процессе рассмотрения команд SQL, предназначенных для физического создания таблиц в базе данных.

Приведенную ниже схему можно найти на сайте компании Postgres Professional по адресам, указанным в начале этого раздела.


Bookings
Бронирования
book_ref
∗ book_date
∗ total_amount
Airports
Аэропорты
airport_code
∗ airport_name
∗ city
∗ longitude
∗ latitude
∗ timezone
Tickets
Билеты
ticket_no
∗ book_ref
∗ passenger_id
∗ passenger_name
○ contact_data
Ticket_flights
Перелеты
ticket_no
flight_id
∗ fare_conditions
∗ amount
Flights
Рейсы
flight_id
∗ flight_no
∗ scheduled_departure
∗ scheduled_arrival
∗ departure_airport
∗ arrival_airport
∗ status
∗ aircraft_code
○ actual_departure
○ actual_arrival
Aircrafts
Самолеты
aircraft_code
∗ model
∗ range
Boarding_passes
Посадочные талоны
ticket_no
flight_id
∗ boarding_no
∗ seat_no
Seats
Места
aircraft_code
seat_no
∗ fare_conditions

Контрольные вопросы и задания

1. Какие группы операторов выделяются в составе языка SQL?
2. Дайте неформальное определение основных понятий реляционной модели данных: отношение, кортеж, атрибут.
3. Для чего нужны внешние ключи в реляционных таблицах?
4. Что такое потенциальный ключ?
5.* Предложите пример избыточного потенциального ключа для одной из таблиц базы данных «Авиаперевозки» и объясните, почему он будет избыточным.
6.* В реализации базы данных «Авиаперевозки» предполагается, что самолеты одной модели могут иметь только одну компоновку салона. Представим, что руководством принято решение о том, что нужно учитывать возможность наличия различных компоновок для каждой модели. Какие таблицы придется модифицировать в таком случае и каким образом? Потребуется ли создавать дополнительные таблицы?

Глава 2

Создание рабочей среды

Прежде чем приступать к непосредственному изучению языка SQL, нужно получить доступ к серверу PostgreSQL. Это можно сделать, например, в компьютерном классе или путем обращения к удаленному серверу через терминал. Однако можно создать рабочую среду для себя и на своем локальном компьютере, установив полную версию СУБД PostgreSQL, т. е. сервер и клиентские программы. В этом случае у вас будет гораздо больше полномочий по настройке и использованию PostgreSQL.

В заключительной части главы мы покажем, как развернуть учебную базу данных «Авиаперевозки», наполненную специально подготовленными правдоподобными данными.

2.1. Установка СУБД

Поскольку настоящее учебное пособие предназначено для изучения языка SQL, а не основ администрирования СУБД PostgreSQL, то мы ограничимся лишь краткими указаниями о том, где найти инструкции по установке.

Начать нужно с выбора того дистрибутива СУБД, который вы хотели бы установить. Вы можете выбрать оригинальный вариант PostgreSQL или тот, который предлагается компанией Postgres Professional. Он называется Postgres Pro и содержит не только все функции и модули, входящие в состав стандартного дистрибутива, но и дополнительные разработки, выполненные в компании Postgres Professional. Для изучения основ языка SQL эти дистрибутивы подходят в равной степени. Однако документация на русском языке включена только в состав Postgres Pro.

После того как вы определитесь с конкретным дистрибутивом СУБД, необходимо выбрать операционную систему. PostgreSQL поддерживает множество систем, в том числе различные версии Linux, а также Windows.

Устанавливать рекомендуется последнюю **стабильную** версию СУБД.

Если вы решили воспользоваться оригинальным дистрибутивом PostgreSQL, то найти инструкции по его установке в различных операционных системах можно по адресу https://www.postgresql.org/download/.

Если же вы остановили свой выбор на дистрибутиве Postgres Pro, тогда следует обратиться сюда: https://postgrespro.ru/products/postgrespro/download/latest.

После установки как PostgreSQL, так и Postgres Pro, в среде Windows придется предпринять дополнительные меры, чтобы использование русского алфавита в интерактивном терминале psql не вызывало проблем. Утилита psql рассматривается в следующем разделе.

В процессе установки будет создана отдельная учетная запись пользователя СУБД с именем postgres. Для изучения настоящего пособия создавать дополнительные учетные записи не требуется.

Установив тот или иной дистрибутив PostgreSQL, нужно научиться запускать сервер баз данных, потому что иначе невозможно работать с данными. Как это сделать, подробно описано в документации в разделе 18.3 «Запуск сервера баз данных». Найти этот раздел можно по адресу https://postgrespro.ru/docs/postgresql/current/server-start.html. При установке СУБД в среде Windows создается служба (service) для автоматического запуска сервера PostgreSQL при загрузке операционной системы.

Завершив работу с сервером, нужно корректно остановить (выключить) его. Порядок действий в такой ситуации описан в документации в разделе 18.5 «Выключение сервера». Найти этот раздел можно по адресу https://postgrespro.ru/docs/postgresql/9.6/server-shutdown.html.

2.2. Программа psql — интерактивный терминал PostgreSQL

Для доступа к серверу баз данных в комплект PostgreSQL входит интерактивный терминал psql. Для его запуска нужно ввести команду

```
psql
```

При запуске утилиты psql в среде Windows возможно некорректное отображение букв русского алфавита. Для устранения этого потребуется в свойствах окна, в котором выполняется psql, изменить шрифт на Lucida Console и с помощью команды `chcp` сменить текущую кодовую страницу на CP1251:

```
chcp 1251
```

В среде утилиты psql можно вводить не только команды языка SQL, но и различные сервисные команды, поддерживаемые самой утилитой.

Для получения краткой справки по всем сервисным командам нужно ввести

```
\?
```

Многие такие команды начинаются с символов «\d». Например, для того чтобы просмотреть список всех таблиц и представлений (views), созданных в той базе данных, к которой вы сейчас подключены, введите команду

```
\dt
```

Если же вас интересует определение (попросту говоря, структура) какой-либо конкретной таблицы базы данных, например, `students`, нужно ввести команду

```
\d students
```

Для получения списка всех SQL-команд нужно выполнить команду

```
\h
```

Для вывода описания конкретной SQL-команды, например, `CREATE TABLE`, нужно сделать так:

```
\h CREATE TABLE
```

Эта утилита позволяет сокращать объем ручного ввода за счет дополнения вводимой команды «силами» psql. Например, при вводе SQL-команды можно использовать клавишу <Tab> для дополнения вводимого ключевого слова команды или имени таблицы базы данных. Например, при вводе команды `CREATE TABLE` можно, введя символы «cr», нажать клавишу <Tab> — psql дополнит это слово до «create». Аналогично можно поступить и со вторым словом: для его ввода достаточно ввести лишь буквы «ta» и нажать клавишу <Tab>. Если вы ввели слишком мало букв для того, чтобы утилита psql могла однозначно идентифицировать ключевое слово, дополнения не произойдет. Но в таком случае вы можете нажать клавишу <Tab> дважды и получить список всех ключевых слов, начинающихся с введенной вами комбинации букв.

2.3. Развертывание учебной базы данных

Завершив установку сервера баз данных, мы можем перейти непосредственно к рассмотрению вопроса о том, как развернуть в вашем кластере PostgreSQL учебную базу данных «Авиаперевозки», подготовленную компанией Postgres Professional.

На сайте компании есть раздел, посвященный этой базе данных, найти его можно по ссылке https://postgrespro.ru/education/demodb. Она предоставляется в трех версиях, отличающихся только объемом данных: самая компактная версия содержит данные за один месяц, версия среднего размера охватывает временной период в три месяца, а самая полная версия включает данные за целый год. Все данные были сгенерированы с помощью специальных алгоритмов, обеспечивающих их «правдоподобность». Мы рекомендуем вам начать с компактной версии базы данных «Авиаперевозки», а после получения некоторого опыта написания SQL-запросов вы установите полную версию и уже на ней сможете лучше «прочувствовать» различные тонкости работы с данными больших объемов, например, оцените влияние индексов на скорость доступа к данным.

В качестве первого шага к развертыванию базы данных нужно скачать ее заархивированную резервную копию по ссылке https://edu.postgrespro.ru/demo_small.zip. Затем необходимо извлечь файл из архива:

```
unzip demo_small.zip
```

Извлеченный файл называется demo_small.sql. Теперь создадим базу данных с именем demo в вашем кластере PostgreSQL. Самый краткий вариант команды будет таким:

```
psql -f demo_small.sql -U postgres
```

Если вы хотите перенаправить вывод сообщений, которые генерирует СУБД в процессе работы, с экрана в файлы, то можно поступить так:

```
psql -f demo_small.sql -U postgres > demo.log 2>demo.err
```

Можно разделить стандартное устройство вывода и стандартное устройство вывода ошибок. Обычные сообщения будут перенаправлены в файл demo.log, а сообщения об ошибках — в файл demo.err. Обратите внимание, что между цифрой 2, обозначающей дескриптор стандартного устройства вывода сообщений об ошибках, и знаком «>», обозначающим переадресацию вывода, не должно быть пробела.

Если вам удобнее собрать все сообщения в один общий файл, тогда нужно сделать так:

```
psql -f demo_small.sql -U postgres > demo.log 2>&1
```

Обратите внимание, что все выражение 2>&1 в конце команды пишется без пробелов. Оно указывает операционной системе, что сообщения об ошибках нужно направить туда же, куда выводятся и обычные сообщения.

Если бы наш SQL-файл был очень большим, тогда можно было бы выполнить команду в фоновом режиме, поставив в конце командной строки символ «&», а за ходом процесса в реальном времени наблюдать с помощью команды `tail`.

```
psql -f demo_small.sql -U postgres > demo.log 2>&1 &
tail -f demo.log
```

Выберите один из предложенных вариантов команды для развертывания базы данных и выполните эту команду.

Все готово! Можно подключаться к новой базе данных:

```
psql -d demo -U postgres
```

Контрольные вопросы и задания

1. Выполните процедуру установки СУБД PostgreSQL в среде выбранной вами операционной системы.
2. Ознакомьтесь с утилитой psql с помощью встроенной справки, а также с помощью справки, вызываемой по команде

   ```
   psql --help
   ```

3. Кроме утилиты psql существуют и другие универсальные программы для работы с сервером баз данных PostgreSQL, например, pgAdmin. Это мощная утилита с графическим интерфейсом.

   Самостоятельно установите программу pgAdmin и изучите основные приемы работы с ней.
4. Разверните учебную базу данных. Попробуйте подключиться к ней с помощью утилиты psql. Для выхода из утилиты используйте команду `\q`.