Оглавление

Предисловие 13

Глава 1. Введение 21
Реляционная модель очень плохо понята 22
Некоторые замечания о терминологии 23
Принципы, а не продукты 25
Обзор оригинальной модели 26
Модель и реализация 35
Свойства отношений 39
Базовые и производные отношения 42
Отношения и переменные-отношения 44
Значения и переменные 46
Заключительные замечания 47
Упражнения 49

Глава 2. Типы и домены 51
Типы и отношения 51
Сравнения на равенство 52
Атомарность значений данных 58
Что такое тип? 62
Скалярные и нескалярные типы 66
Скалярные типы в SQL 68
Проверка и приведение типов в SQL 70
Схемы упорядочения в SQL 72
Тип строки и таблицы в SQL 74
Заключительные замечания 76
Упражнения 77

Глава 3. Кортежи и отношения, строки и таблицы 81
Что такое кортеж? 81
Следствия из определений 84
Строки в SQL 86
Что такое отношение? 88
Отношения и их тела 90
Отношения n-мерны 91

Сравнение отношений ... 92
TABLE_DUM и TABLE_DEE ... 93
Таблицы в SQL ... 94
Именование столбцов в SQL ... 96
Заключительные замечания ... 98
Упражнения ... 99

Глава 4. Нет дубликатам, нет null-значениям ... 101
Чем плохи дубликаты? ... 101
Дубликаты: новые проблемы ... 106
Как избежать дубликатов в SQL ... 107
Чем плохи null-значения? ... 109
Как избежать null-значений в SQL ... 113
Замечание о внешнем соединении ... 115
Заключительные замечания ... 116
Упражнения ... 117

Глава 5. Базовые переменные-отношения, базовые таблицы ... 121
Определения данных ... 122
Обновление – это операция над множеством ... 122
Реляционное присваивание ... 125
Принцип присваивания ... 126
Еще о потенциальных ключах ... 127
Еще о внешних ключах ... 130
Переменные-отношения и предикаты ... 134
Отношения и типы ... 137
Упражнения ... 139

Глава 6. SQL и реляционная алгебра I: оригинальные операторы ... 142
Предварительные сведения ... 142
Еще о замкнутости ... 145
Ограничение ... 149
Проекция ... 149
Соединение ... 151
Объединение, пересечение и разность ... 155
Какие операторы являются примитивными? ... 157
Пошаговое конструирование выражений ... 157
В чем смысл реляционных выражений? ... 159
Вычисление табличных выражений в SQL ... 160
Трансформация выражений ... 161
Зависимость от имен атрибутов ... 165
Упражнения ... 167

Глава 7. SQL и реляционная алгебра II: дополнительные операции ... 171
Полусоединение и полуразность ... 171
Расширение ... 172
Отношения-образы ... 174
Деление ... 177
Агрегатные операторы ... 179
Еще об отношениях-образах ... 183
Обобщение ... 185
Еще об обобщении ... 190
Группирование и разгруппирование ... 191
Запросы «что если» ... 193
А как насчет ORDER BY? ... 194
Упражнения ... 195

Глава 8. SQL и ограничения целостности ... 200
Ограничения типа ... 201
Еще об ограничениях типа ... 204
Ограничения типа в SQL ... 205
Ограничения базы данных ... 206
Ограничения базы данных в SQL ... 210
Транзакции ... 211
Почему ограничения базы данных должны проверяться немедленно ... 213
Но разве можно не откладывать проверку некоторых ограничений? ... 216
Ограничения и предикаты ... 219
Разное ... 221
Упражнения ... 223

Глава 9. SQL и представления ... 228
Представления – это переменные-отношения ... 229
Представления и предикаты ... 233
Операции выборки ... 234
Представления и ограничения ... 236
Операции обновления ... 240
Зачем нужны представления? ... 244
Взгляды и снимки ... 245
Упражнения ... 247

Глава 10. SQL и формальная логика ... 250
Простые и составные высказывания ... 251
Простые и составные предикаты ... 254

Квантификация ... 256
Реляционное исчисление ... 260
Еще о квантификации ... 267
Некоторые эквиваленции ... 274
Заключительные замечания ... 277
Упражнения ... 278

Глава 11. Использование формальной логики для формулирования SQL-выражений ... 281
Некоторые правила трансформации ... 282
Пример 1. Логическая импликация ... 284
Пример 2. Добавление квантора всеобщности ... 285
Пример 3. Импликация и квантор всеобщности ... 286
Пример 4. Коррелированные подзапросы ... 288
Пример 5. Именование подвыражений ... 290
Пример 6. Еще об именовании подвыражений ... 293
Пример 7. Устранение неоднозначности ... 294
Пример 8. Использование COUNT ... 296
Пример 9. Запросы с соединением ... 297
Пример 10. Квантор UNIQUE ... 298
Пример 11. Сравнения с ALL или ANY ... 299
Пример 12. GROUP BY и HAVING ... 303
Упражнения ... 304

Глава 12. Различные вопросы, связанные с SQL ... 306
SELECT * ... 307
Явные таблицы ... 307
Квалификация имен ... 307
Переменные кортежа ... 308
Подзапросы ... 311
«Потенциально недетерминированные» выражения ... 314
Пустые множества ... 315
БНФ-грамматика табличных выражений SQL ... 315
Упражнения ... 318

Приложение А. Реляционная модель ... 320
Реляционная и другие модели ... 322
Определение реляционной модели ... 325
Цели реляционной модели ... 331
Некоторые принципы баз данных ... 331
Что осталось сделать? ... 333

Приложение В. Теория проектирования баз данных ... 338
Место теории проектирования ... 339

Функциональные зависимости
и нормальная форма Бойса/Кодда 342
Зависимости соединения и пятая нормальная форма 348
Тост за здоровье нормализации 357
Ортогональность 361
Некоторые замечания о физическом проектировании 364
Заключительные замечания 366
Упражнения 368

Приложение C. Ответы к упражнениям 372
Глава 1 372
Глава 2 379
Глава 3 387
Глава 4 392
Глава 5 398
Глава 6 404
Глава 7 413
Глава 8 424
Глава 9 433
Глава 10 440
Глава 11 448
Глава 12 450
Приложение B 450

Приложение D. Дополнительная литература 460

Алфавитный указатель 469

Предисловие

Язык SQL распространен повсеместно. Но работать с SQL трудно: он сложен, запутан, при написании SQL-команд легко допустить ошибку – рискну предположить, что куда легче, чем можно судить по уверениям апологетов этого языка. Поэтому, чтобы уверенно писать правильный SQL-код (то есть такой, который делает в точности то, что вам нужно, не больше и не меньше), необходимо четко следовать некоторому правилу. А основной тезис настоящей книги заключается в том, что таким правилом может стать *использование SQL в соответствии с реляционной теорией*. Но что это означает? Разве SQL не является изначально реляционным языком?

Да, конечно, SQL – стандартный язык для всех реляционных баз данных, но сам по себе этот факт не делает его реляционным. Как это ни печально, SQL слишком часто отходит от принципов реляционной теории; строки-дубликаты и null-значения – два наиболее очевидных примера, которыми проблема отнюдь не ограничивается. Следовательно, он предлагает вам путь, который может привести в западню. А если вы не желаете попадать в западню, то должны понимать реляционную теорию (что она собой представляет и для чего предназначена), знать, в чем именно SQL отклоняется от этой теории, и уметь избегать проблем, которые могут из этого проистекать. Одним словом, SQL следует использовать в реляционном духе. Тогда вы сможете действовать так, будто SQL на самом деле реляционный язык, и получать все преимущества от работы с тем, что по сути является истинно реляционной системой.

В подобной книге не возникло бы необходимости, если бы все и без того использовали SQL реляционным образом, – но, увы, это не так. Напротив, в современном применении SQL я вижу проявление множества вредных тенденций. Более того, эти способы применения относятся к рекомендуемым в различных учебниках и иных публикациях авторами, которые должны были бы относиться к своей работе более ответственно (имен не привожу и на посмешище выставлять не хочу); анализ литературы в этом отношении оставляет удручающее впечатление. Реляционная модель появилась на свет в 1969 году, и вот – почти сорок лет спустя – она, похоже, так и не понята по-настоящему сообществом пользователей СУБД в целом. Отчасти поэтому в основу организации настоящей книги положена сама реляционная модель; подробно объясняются различные аспекты этой модели, и для каждого аспекта демонстрируется, как лучше всего реализовать его средствами SQL.

Предварительные требования

Я предполагаю, что вы применяете СУБД в своей работе, поэтому уже в какой-то мере знакомы с языком SQL. Точнее, я считаю, что вы имеете практическое представление либо о стандарте SQL, либо (что более вероятно) о каком-либо продукте, где применяется SQL. Однако я не рассчитываю на глубокое знание реляционной теории как таковой (хотя надеюсь, что вы все же согласны с тем, что реляционная теория – вещь хорошая, и по возможности ее следует придерживаться). Поэтому во избежание недопонимания я буду подробно описывать различные свойства реляционной модели, а также показывать, как SQL согласуется с этими свойствами. Однако я не стану пытаться обосновывать существование этих свойств, а буду предполагать, что вы достаточно подкованы в тематике баз данных, чтобы понимать, к примеру, зачем используется понятие ключа, или почему иногда приходится выполнять операцию соединения, или зачем требуется поддержка отношений многие-ко-многим. (Если бы я решил включать все определения и обоснования, то получилась бы совсем другая книга – гораздо больше по размеру, не говоря уже обо всем остальном; да и в любом случае такая книга уже написана.)

Я сказал, что ожидаю от вас достаточно близкого знакомства с SQL. Добавлю, однако, что некоторые аспекты SQL я все равно буду объяснять подробно – особенно те, что на практике применяются нечасто. (В качестве примера упомяну «потенциально недетерминированные выражения». См. главу 12.)

«Database in Depth»

Эта книга базируется на ранее изданной книге «Database in Depth: Relational Theory for Practitioners» (O'Reilly, 2005) и должна заменить ее. В предыдущей книге я ставил перед собой такую цель (цитата взята из предисловия):

> Посвятив много лет работе с базами данных в разном качестве, я пришел к выводу, что существует настоятельная потребность в книге для практиков (не новичков), где принципы реляционной теории излагались бы вне связи с особенностями конкретных существующих продуктов, коммерческими обычаями или стандартом SQL. Таким образом, в качестве читательской аудитории я вижу опытных пользователей СУБД, достаточно честных, чтобы сознаться в том, что они не понимают теоретических основ той области, в которой работают, в той мере, в какой могли бы и должны были бы понимать. Этой теорией, естественно, является реляционная модель, и хотя фундаментальные идеи, положенные в ее основу, очень просты, надо признать, что они чуть ли не повсеместно неверно представляются, или недооцениваются,

> или то и другое вместе. Фактически часто их не понимают вовсе. Вот, к примеру, несколько вопросов по реляционной теории[1]... На сколько из них вы сможете ответить?
>
> Что в точности означает первая нормальная форма?
>
> Какая связь существует между отношениями и предикатами?
>
> Что такое семантическая оптимизация?
>
> Что такое отношение-образ?
>
> Почему так важна полуразность?
>
> Почему отложенная проверка ограничений целостности не имеет смысла?
>
> Что такое переменная-отношение?
>
> Что такое предваренная нормальная форма?
>
> Может ли отношение иметь атрибут, значениями которого являются отношения?
>
> Является ли язык SQL реляционно полным?
>
> Почему так важен *принцип информации*?
>
> Как XML укладывается в реляционную модель?
>
> Настоящая книга дает ответы на эти и многие другие вопросы. В общем и целом, ее цель – помочь пользователям-практикам баз данных глубоко разобраться в реляционной теории и применить полученные знания в повседневной профессиональной деятельности.

Как следует из последнего предложения, я надеялся, что читатели той книги смогут применить изложенные идеи в своей работе без дальнейшей помощи с моей стороны. Но с тех пор я осознал, что вопреки устоявшемуся мнению язык SQL настолько труден, что вопрос о том, как применять его, не нарушая реляционных принципов, далеко не праздный. Поэтому я решил дополнить первоначальную книгу, включив в нее явные, конкретные рекомендации именно в этом направлении (то есть как использовать SQL в реляционном духе). Таким образом, в этой книге я преследовал ту же цель, что и раньше (помочь пользователям-практикам баз данных глубоко разобраться в реляционной теории и применить полученные знания в повседневной профессиональной деятельности), но постарался представить материал в виде, быть может, более удобном для усвоения и уж точно – для применения. Иными словами, я включил много материала, относящегося конкретно к языку SQL (и именно поэтому размер так сильно увеличился по сравнению с предыдущим вариантом).

---

[1] По причинам, которые здесь несущественны, я заменил несколько вопросов из оригинального перечня.

Дополнительные замечания о тексте

Я должен высказать еще несколько предварительных замечаний. Прежде всего, мое собственное понимание реляционной модели с годами развивалось. В этой книге изложены мои нынешние представления о предмете; поэтому если вы обнаружите технические расхождения – а они есть – между этой и другими моими книгами (в частности, и той, которую эта призвана заменить), правильным следует считать написанное здесь. Впрочем, сразу оговорюсь, что расхождения по большей части не слишком существенны; более того, я всегда соотношу новые термины и понятия со старыми, если считаю, что в этом есть необходимость.

Во-вторых, я, как и обещано, собираюсь говорить о теории, но опираясь на свою уверенность, что *нет ничего практичнее хорошей теории*. Я специально выделил этот момент, потому что многие, похоже, придерживаются противоположного мнения и считают: все, что отдает теоретизированием, на практике неприменимо. Но истина в том, что теория (по крайней мере, реляционная, а именно о ней я и веду речь) очень даже приближена к практике. Она создавалась как теория *не* ради теории, а ради построения систем, на все сто процентов практических. Каждая деталь этой теории обусловлена весьма практическими соображениями. Один из рецензентов предыдущей книги, Стефан Фарульт (Stéphane Faroult), писал: «Приобретя некоторый практический опыт, вы начинаете осознавать, что без знания теории не обойтись». Более того, эта теория не только практична, она еще и фундаментальна, проста, понятна, полезна, а иногда еще и *забавна* (как я надеюсь продемонстрировать в этой книге).

Разумеется, за самой яркой иллюстрацией вышеприведенного тезиса не нужно ходить дальше самой реляционной модели. Вообще вряд ли необходимо отстаивать мнение о практичности теории в том контексте, который мы имеем: существование многомиллиардной индустрии, целиком основанной на одной замечательной теоретической идее. Однако я предвижу и такую циничную позицию: «Ну ладно, а что эта теория сделала лично для меня в последнее время?» Иными словами, те из нас, кто действительно убежден в важности теории, должны постоянно противостоять критикам – и это еще одна причина, по которой я считаю эту книгу полезной.

В-третьих, как я уже говорил, в этой книге детально излагаются различные аспекты SQL и реляционной модели. (Я сознательно почти не затрагивал темы, не имеющие отношения собственно к реляционной теории, в частности транзакции.) Я всюду старался ясно отмечать, когда обсуждение относится только к SQL, когда – только к реляционной модели, а когда – к тому и другому. Однако должен подчеркнуть, что не стремился к исчерпывающему рассмотрению всех деталей SQL. Язык SQL очень сложен и предоставляет много разных способов решения одной

и той же задачи, в нем так много исключений и особых случаев, что попытка охватить все – даже если бы это было возможно, в чем я лично сомневаюсь, – оказалась бы непродуктивной, а книга при этом выросла бы до необозримых размеров. Поэтому я старался уделить внимание тому, что считаю действительно важным, не растекаясь при этом мыслью по древу. Хотелось бы заявить, что если вы будете делать все, что я советую, и не будете делать того, что я не советую, то это станет первым приближением к безопасной работе: вы будете использовать SQL реляционно. Но только вам судить, насколько это заявление оправдано и оправдано ли вообще.

Ко всему сказанному следует добавить, что, к сожалению, существуют ситуации, когда SQL невозможно использовать реляционно. Например, проверку некоторых ограничений целостности приходится откладывать (обычно, чтобы выиграть время), хотя реляционная модель считает такую отложенную проверку логической ошибкой. В этой книге приводятся рекомендации о том, как поступать в подобных случаях, но боюсь, что по большей части они сводятся к тривиальному совету: *делайте так, как считаете наиболее правильным*. Надеюсь, по крайней мере, что вы будете понимать, в чем состоит опасность отклонения от модели.

Должен также сказать, что некоторые предлагаемые рекомендации не относятся к реляционной теории, а имеют общий характер, хотя иногда у них есть и «реляционные» последствия (не всегда очевидные, кстати говоря). Хорошим примером может служить совет *избегать приведения типов*.

В-четвертых, обратите внимание, что под словом *SQL* я в этой книге понимаю исключительно стандартную версию языка, а не какой-то конкретный диалект (если только противное не оговорено явно). В частности, в согласии со стандартом я подразумеваю произношение «эс кью эл», а не «сиквел»[1] (хотя на практике последнее широко распространено).

В-пятых, эту книгу следует читать в основном последовательно за немногими исключениями, оговоренными здесь и далее в самом тексте (большинство глав в той или иной мере зависят от ранее изложенного материала, поэтому старайтесь не перескакивать из одного места в другое). Кроме того, в каждую главу включены упражнения. Конечно, выполнять их необязательно, но мне кажется, что было бы разумно попробовать свои силы хотя бы на некоторых. Ответы, в которых часто содержится дополнительная информация по теме, приведены в приложении С.

И наконец, хотелось бы упомянуть о некоторых видеосеминарах, основанных на материалах этой книги. Дополнительную информацию см. по адресу *http://www.clik.to/chris_date* или *http://www.thethirdmanifesto.com*.

[1] Поэтому мы пишем «об SQL», а не «о SQL». – *Прим. перев.*

Типографские соглашения

В книге применяются следующие соглашения:

Курсив

Служит для визуального выделения. Этим шрифтом обозначаются новые термины. Кроме того, он применяется в основном тексте, когда нужно сказать, что вместо чего-то следует подставить некоторую переменную, например x.

`Моноширинный шрифт`

Применяется для примеров кода.

`Моноширинный курсив`

В примерах кода обозначает переменную или значение, вводимое пользователем.

О примерах кода

Эта книга призвана помогать вам в работе. Поэтому вы можете использовать приведенный в ней код в собственных программах и в документации. Спрашивать у нас разрешение необязательно, если только вы не собираетесь воспроизводить значительную часть кода. Например, не требуется разрешение, чтобы включить в свою программу несколько фрагментов кода из книги. Однако для продажи или распространения примеров на компакт-диске нужно получить разрешение. Можно без ограничений цитировать книгу и примеры в ответах на вопросы. Но чтобы включить значительные объемы кода в документацию по собственному продукту, нужно получить разрешение.

Мы высоко ценим, хотя и не требуем, ссылки на наши издания. В ссылке обычно указываются название книги, имя автора, издательство и ISBN, например: «SQL and Relational Theory, C. J. Date», Copyright 2009 C. J. Date, 978-0-596-52306-0.

Если вы полагаете, что планируемое использование кода выходит за рамки изложенной выше лицензии, пожалуйста, обратитесь к нам по адресу *permissions@oreilly.com*.

Замечания и вопросы

Я очень старался, чтобы в книге не было ошибок, но что-то мог упустить. Если вы найдете какой-то огрех, просьба уведомить издательство по адресу:

O'Reilly Media, Inc.
1005 Gravenstein Highway North
Sebastopol, CA 95472

800-998-9938 (для США и Канады)
707-829-0515 (международный или местный)
707-829-0104 (факс)

Можно также посылать сообщения по электронной почте. Чтобы подписаться на рассылку или заказать каталог, отправьте письмо на адрес:

info@oreilly.com

Замечания и вопросы технического характера следует отправлять по адресу:

bookquestions@oreilly.com

Для этой книги создана веб-страница, на которой выкладываются примеры и списки замеченных ошибок (ранее отправленные извещения об ошибках и исправления доступны для всеобщего обозрения). Адрес страницы:

http://www.oreilly.com/catalog/9780596523060

Дополнительную информацию об этой и других книгах можно найти на сайте издательства O'Reilly:

http://www.oreilly.com

Доступность на Safari

Если на обложке вашей любимой книги присутствует значок Safari® Enabled, это означает, что книга доступна в сетевой библиотеке Safari издательства O'Reilly.

У Safari есть преимущество перед обычными электронными книгами. Это виртуальная библиотека, которая позволяет легко находить тысячи технических книг, копировать примеры программ, загружать отдельные главы и быстро получать точную и актуальную информацию. Бесплатный доступ по адресу *http://safari.oreilly.com*.

Благодарности

О том, чтобы переработать предыдущую книгу, включив в нее дополнительный материал по SQL, я подумывал довольно давно, но последним толчком, заставившим меня наконец засесть за эту работу, явилось мое присутствие в 2007 году на одном курсе, предназначенном для практических пользователей СУБД. Курс вел Тоон Коппелаарс (Toon Koppelaars) на базе книги, которую он написал совместно с Лексом де Хааном (Lex de Haan) (рецензентом этой книги), очень неплохой, кстати. Но поразительным оказалось то, насколько беспомощно выглядели попытки слушателей применить реляционные и логические принципы к привычному для них использованию SQL. Конечно, у слушателей были какие-то знания по этой теме – в конце концов, все они рабо-

тали с базами данных на практике, – но мне показалось, что их определенно нужно направить в сторону применения этих идей в повседневной деятельности. Поэтому я и написал эту книгу. И потому я благодарен в первую очередь Тоону и Лексу, которые придали импульс, необходимый для начала работы над проектом. Я благодарен также рецензентам Хербу Эдельштейну (Herb Edelstein), Шеери Ктитцеру (Sheeri Ktitzer), Энди Ораму (Andy Oram), Питеру Робсону (Peter Robson) и Бэрону Шварцу (Baron Schwartz) за замечания по первым черновикам рукописи и Хью Дарвену (Hugh Darwen) и Джиму Мелтону (Jim Melton) за техническую помощь иного характера. Кроме того, я как всегда благодарен своей жене Линди за неизменную многолетнюю поддержку этого и прочих моих проектов. Наконец, спасибо всему коллективу издательства O'Reilly и особенно Изабель Канкл (Isabel Kunkle), Энди Ораму (Andy Oram) и Адаму Уитверу (Adam Witwer) за ценные советы и за то, что они ободряли и поддерживали меня на протяжении всего времени работы над книгой.

К. Дж. Дейт

Хилдсбург, штат Калифорния,
2008

1

Введение

Реляционный подход к SQL – вот тема или одна из тем настоящей книги. Разумеется, чтобы должным образом раскрыть эту тему, мне придется рассматривать вопросы, касающиеся как реляционной теории, так и самого языка SQL. И хотя это замечание очевидным образом применимо ко всей книге, к первой главе оно относится в особенности. Поэтому здесь мы почти не будем говорить о языке SQL как таковом. Моя цель сейчас – дать обзор материала, большая часть которого вам, скорее всего, уже известна. Я хочу обозначить некую отправную точку, то есть заложить фундамент, на котором можно будет возводить здание книги. Но, даже искренне рассчитывая на то, что вы в основном знакомы с тем, о чем я собираюсь рассказать в этой главе, я все же предлагаю не пропускать ее. Вы должны знать то, что знать надлежит (надеюсь, вы понимаете, что я хочу сказать); в частности, вы должны владеть всем, что понадобится для понимания материала следующих глав. На самом деле я – со всем уважением – порекомендовал бы вообще не пропускать рассмотрение тем, которые вам кажутся знакомыми. Например, так ли вы уверены, что знаете, что в реляционной терминологии понимается под ключом? Или что такое соединение?[1]

---

[1] По крайней мере один ученый муж этого не понимает. Следующий отрывок взят из документа, который (как и эта книга!) имеет целью наставить пользователей SQL: «Не применяйте соединения... Подход Oracle и SQL Server к этой концепции принципиально различен... Вы можете получить неожиданные результирующие наборы... Необходимо понимать основные типы соединений... Экви-соединение строится путем выборки всех данных двух отдельных источников и создания из них одной большой таблицы... В случае внутреннего соединения две таблицы соединяются по внутренним столбцам... В случае внешнего соединения две таблицы соединяются по внешним столбцам... (*да-да, именно так написано в оригинале. – Прим. перев.*). В случае левостороннего соединения две таблицы соединяются по левым столбцам. В случае правостороннего соединения две таблицы соединяются по правым столбцам».

Реляционная модель очень плохо понята

Профессионалы в любой области должны знать лежащие в ее основе фундаментальные идеи. Поэтому профессионал в области баз данных должен знать реляционную теорию, поскольку именно реляционная модель является фундаментом (уж во всяком случае здоровенной частью фундамента), на котором покоится вся эта отрасль. В наше время любой курс по управлению базами данных, академический или коммерческий, на словах привержен идее преподавания реляционной модели, но в большинстве случаев преподавание поставлено очень плохо, если судить по результатам; безусловно, в сообществе пользователей баз данных эта модель недопонята. Перечислим некоторые возможные причины такого положения вещей.

- Модель преподносится в отрыве от всего, в «вакууме». Поэтому начинающему очень трудно уловить значимость материала или понять, какие проблемы модель призвана решить, или то и другое одновременно.
- Сами преподаватели не до конца понимают или не в состоянии оценить важность материала.
- Чаще всего на практике модель как таковая не рассматривается вовсе – вместо нее преподается язык SQL или какой-то его диалект, например принятый в Oracle.

Поэтому эта книга обращена к тем людям, которые на практике работают с базами данных и в особенности с языком SQL и каким-то боком связаны с реляционной моделью, но знают о ней не так много, как должны или хотели бы знать. Она определенно *не рассчитана* на начинающих, однако не является курсом повышения квалификации. Конкретно, я уверен, что вы что-то знаете о языке SQL, но – не сочтите за обиду – если ваши знания о реляционной модели проистекают исключительно из знания SQL, то, боюсь, вы не понимаете ее так хорошо, как следовало бы, и очень может статься, что какие-то «ваши знания неверны». Не устану повторять: *SQL и реляционная модель – вовсе не одно и то же*. В качестве иллюстрации приведу некоторые вопросы реляционной теории, которые в SQL трактуются недостаточно ясно (и это еще мягко сказано):

- Что в действительности представляют собой базы данных, отношения и кортежи
- В чем разница между отношениями-значениями и переменными-отношениями
- Значимость предикатов и высказываний
- Важность имен атрибутов
- Важнейшая роль ограничений целостности

и так далее (список далеко не полный). Все эти и многие другие вопросы рассматриваются в настоящей книге.

Еще раз повторю: если ваши знания о реляционной модели проистекают исключительно из знания SQL, то ваши знания могут оказаться неверными. Отсюда, в частности, следует, что, читая эту книгу, вы, возможно, обнаружите, что кое-что из уже известного следует забыть, а переучиваться, к сожалению, всегда трудно.

Некоторые замечания о терминологии

Вероятно, вы сразу же обратили внимание на то, что в перечне вопросов реляционной теории из предыдущего раздела я употребил формальные термины «отношение», «кортеж» и «атрибут». Но в SQL эти термины не используются, там применяются более «дружественные» слова: таблицы, строка и столбец. Вообще говоря, я с пониманием отношусь к идее употребления понятных пользователю слов, если это помогает усвоению идей. Но в данном случае мне кажется, что как раз усвоению идей такая терминология не способствует – как это ни печально; напротив, она лишь искажает суть и оказывает дурную услугу пониманию истинного смысла.

А истина заключается в том, что отношение – это не таблица, кортеж – не строка, а атрибут – не столбец. И хотя в неформальном контексте такое словоупотребление может показаться приемлемым – я и сам так часто говорю, – я настаиваю, что приемлемо оно лишь в том случае, когда мы ясно понимаем, что эти дружественные термины – не более чем приближение к истине, они совершенно не ухватывают смысл того, что происходит на самом деле. Скажу по-другому: если вы понимаете истинное положение вещей, то благоразумное употребление дружественных терминов может оказаться здравой идеей, но, чтобы понять и оценить это истинное положение, необходимо освоить более формальную терминологию. Поэтому в этой книге я буду, как правило, пользоваться формальными терминами – по крайней мере тогда, когда говорю о реляционной модели, противопоставляя ее SQL, – и в нужном месте приведу их строгие определения. Напротив, в контексте SQL я буду употреблять присущую ему терминологию.

И еще одно замечание о терминологии: упомянув, что SQL пытается упростить один набор терминов, я должен добавить, что он сделал все возможное для усложнения другого. Я имею в виду использование терминов *оператор*, *функция*, *процедура*, *подпрограмма* и *метод*; все они обозначают по существу одно и то же (быть может, с незначительными вариациями). В этой книге я всюду буду употреблять слово *оператор*.

Раз уж зашла речь об SQL, позвольте напомнить, что (как отмечалось в предисловии) под этим я понимаю исключительно стандартную вер-

сию языка[1], если не считать нескольких мест, где по контексту требуется иное.

- Иногда я употребляю терминологию, отличающуюся от стандартной. Например, я пишу *табличное выражение* (table expression) вместо принятого в стандарте *выражение запроса* (query expression), потому что (а) значением такого выражения является таблица, а не запрос, и (б) запросы – не единственный контекст, в котором такие выражения используются. (На самом деле, в стандарте применяется термин *табличное выражение*, но в гораздо более узком смысле; конкретно, он обозначает то, что следует за фразой SELECT в выражении SELECT.)
- Продолжая предыдущую мысль, должен добавить, что не все табличные выражения допустимы в SQL в любом контексте, где их использование можно было бы предположить. В частности, результат явной операции JOIN, хотя он, безусловно, обозначает таблицу, не может встречаться в качестве «отдельно стоящего» табличного выражения (то есть на самом внешнем уровне вложенности), а также не может выступать в качестве табличного выражения в скобках, которое составляет подзапрос (см. главу 12). *Обратите внимание, что подобные замечания применимы ко многим обсуждениям в основном тексте книги, но повторять их каждый раз было бы скучно, поэтому я этого делать не буду.* (Однако все это сведено в БНФ-грамматике в главе 12.)
- Я игнорирую те аспекты стандарта, которые можно считать чрезмерно эзотерическими, особенно если они не являются частью того, что в стандарте называется Core SQL (Базовый SQL), или имеют мало общего с реляционной обработкой как таковой. В качестве примеров упомяну так называемые аналитические или оконные функции (OLAP), динамический SQL, рекурсивные запросы, временные таблицы и детали типов, определенных пользователем.
- По причинам, отчасти связанным с типографским набором, я применяю для комментариев стиль, отличающийся от стандартного. Точнее, комментарии печатаются курсивом и обрамлены ограничителями «/*» и «*/».

Не забывайте, что все реализации SQL включают средства, не описанные в стандарте. Типичный пример – идентификаторы строк. Моя общая рекомендация относительно таких средств такова: пользуйтесь ради бога, но только если они не нарушают реляционных принципов (в конце концов, эта книга посвящена описанию *реляционного* подхода к SQL). Например, применение идентификаторов строк, скорее всего, нарушит так называемый *принцип взаимозаменяемости* (см. гла-

[1] International Organization for Standardization (ISO): *Database Language SQL*, Document ISO/IEC 9075:2003 (2003).

ву 9), и если это так, я бы точно не стал их использовать. Но и в этом, и во всех остальных случаях действует универсальное правило: можете делать все, что хотите, если только знаете, что делаете.

Принципы, а не продукты

Стоит потратить немного времени на вопрос о том, почему профессионалу в области баз данных необходимо (как я уже отмечал выше) знать реляционную модель. Причина в том, что реляционная модель не связана ни с каким конкретным продуктом; она имеет дело только с принципами. Что я понимаю под принципами? Словарь дает такие определения:

> **Принцип** – основное, исходное положение теории, учения, мировоззрения; убеждение, взгляд на вещи; основная особенность в устройстве чего-либо.[1]

Важнейшая особенность принципов состоит в их долговечности. Продукты и технологии (и язык SQL в том числе) все время изменяются – принципы остаются постоянными. Допустим, к примеру, что вы знаете СУБД Oracle; предположим даже, что вы крупный специалист по Oracle. Но если ваши знания ограничены только Oracle, то они могут оказаться непригодными, скажем, в среде DB2 или SQL Server (а могут даже затруднить освоение новой среды). Однако если вы знаете основополагающие принципы – иными словами, реляционную модель, – ваши знания и навыки *переносимы*: вы сможете применить их в любой среде, и они никогда не устареют.

Поэтому в этой книге мы будем иметь дело с принципами, а не с продуктами, с основами, а не с преходящими увлечениями. Но я понимаю, что в реальном мире иногда приходится идти на компромиссы. Например, иногда имеются веские причины проектировать базу данных способом, далеким от теоретически оптимального. В качестве другого примера снова обратимся к SQL. Хотя нет сомнений, что SQL можно использовать реляционно (по крайней мере, в большинстве случаев), иногда обнаруживается – ведь существующие реализации так далеки от совершенства, – что это влечет за собой существенное падение производительности... и тогда вы более или менее вынуждены делать что-то не «истинно реляционное» (например, писать запрос неестественным образом, чтобы заставить реализацию воспользоваться индексом). Однако я абсолютно убежден, что такие компромиссы всегда следует оценивать с *концептуальных позиций*. Это означает, что:

- Идя на такой компромисс, вы должны понимать, что делаете.
- Вы должны понимать, как выглядит теоретически правильное решение, и иметь основательные причины для отхода от него.

---

[1] Перевод дан по изданию Толкового словаря русского языка С. И. Ожегова. – *Прим. перев.*

- Вы должны документировать эти причины; тогда в случае, если в будущем они отпадут (например, из-за того, что в новой версии продукта какая-то функция реализована более удачно), можно будет отказаться от первоначального компромисса.

Следующий афоризм – который приписывается Леонардо да Винчи (1452–1519), и, стало быть, ему уже около 500 лет – великолепно описывает эту ситуацию:

> Всякий, кто полагается на практику, не зная теории, подобен кормчему, вступающему на судно без руля и компаса, – он не знает, куда плывет. *Практика всегда должна опираться на твердые теоретические основания.*

(Курсив добавлен мной.)

Обзор оригинальной модели

В этом разделе я хочу задать отправную точку для последующего обсуждения; здесь приводится обзор некоторых базовых аспектов реляционной модели в том виде, в котором она была определена изначально. Обратите внимание на уточнение – «была определена изначально»! Бытует широко распространенное заблуждение, будто реляционная модель – вещь абсолютно неизменная. Это не так. В этом отношении она напоминает математику: математика тоже не статична, а изменяется со временем. Да ведь и сама реляционная модель – это тоже отрасль математики и как таковая эволюционирует по мере доказательства новых теорем и получения новых результатов. Более того, новый вклад может быть внесен любым компетентным специалистом. И, подобно математике, реляционная модель, хотя и была первоначально изобретена одним человеком, ныне стала плодом совместных усилий и принадлежит всему человечеству.

Кстати говоря, если вы не в курсе, этим человеком был Э. Ф. Кодд, в то время работавший исследователем в корпорации IBM (Э – это Эдгар, Ф – Фрэнк, но он всегда подписывался одними инициалами; а для друзей, к которым я с гордостью отношу и себя, он был просто Тедом). В конце 1968 года Кодд, математик по образованию, впервые понял, как применить математику для закладывания строгих принципов в основание области знания – управления базами данных, – которой в то время таких качеств остро недоставало. Его первоначальное определение реляционной модели появилось в научно-исследовательском отчете IBM в 1969 году, и я еще вернусь к этой работе в приложении D.

Структурные свойства

В оригинальной модели было три основных компонента – структура, целостность и средства манипулирования, и я коротко опишу каждый из них. Но сразу прошу учесть, что все даваемые мной «определе-