Содержание

ВВЕДЕНИЕ

SQL (обычно произносится "SEQUEL") — структурированный язык запросов (Structured Query Language). Он позволяет создавать *реляционные* базы данных, представляющие собой набор связанных данных, хранящихся в таблицах, и оперировать ими.

Мир баз данных имеет тенденцию к постоянной интеграции, приведшей к необходимости разработки стандартного языка, пригодного для использования на множестве современных компьютерных платформ. Стандартный язык дает возможность пользователям освоить один набор команд и применять его для создания, поиска, изменения и передачи данных независимо от того, работает ли он на персональном компьютере, на рабочей станции или на большой вычислительной машине. В компьютерном мире пользователь, владеющий таким языком, имеет огромные возможности по применению и интеграции информации из множества разнообразных источников.

Благодаря своей элегантности и независимости от специфики компьютера, а также поддержке лидерами в области технологии реляционных баз данных, SQL стал и в ближайшем обозримом будущем останется таким стандартным языком. Именно по этой причине, тот, кто предполагает работать с базами данных в девяностые годы нашего столетия, должен владеть языком SQL.

Стандарт SQL определен американским национальным институтом стандартов (American National Standarts Institute) и в настоящее время принят также ISO (International Standards Organization) в качестве международного стандарта. Однако подавляющее большинство коммерческих программ, связанных с обработкой баз данных, расширяет возможности SQL за рамки того, что определено ANSI, добавляя полезные новые черты. Правда, иногда они нарушают стандарт в худшую сторону, тогда как хорошие идеи имеют тенденцию повторяться и становятся стандартом "де факто" или "рыночным" стандартом. В этой книге материал представлен в соответствии с ANSI-стандартом с учетом наиболее общих отклонений от него. Для того, чтобы обнаружить отличия от стандарта, можно воспользоваться документацией по программному обеспечению.

Кто может воспользоваться этой книгой?

Для чтения этой книги требуются минимальные знания из области компьютеров и баз данных. Использовать SQL проще, чем многие другие, менее компактные языки, поскольку при работе на SQL не определяются процедуры, необходимые для получения желаемого результата. Эта книга вводит в мир языка SQL последовательно, содержит множество примеров и упражнений к каждой главе, цель которых — отточить понима-

ние материала и мастерство. Можно выполнять полезные задания немедленно, и, по мере их выполнения, мастерство будет расти.

Поскольку SQL является частью многих программ, выполняющихся на различных компьютерах, никаких предположений относительно специфики использования языка не делается. Эта книга является самым общим пособием. Вы сможете непосредственно применить полученные знания в любой системе, использующей SQL.

Книга предназначена для новичков в области баз данных, однако SQL представлен в ней достаточно глубоко. Примеры отражают множество ситуаций, возникающих в реальных деловых областях приложения. Некоторые из них достаточно сложны, так как приводятся с целью показать все возможные варианты применения SQL.

Как организована эта книга?

Каждая глава вводит новую группу взаимосвязанных понятий и определений. Они базируются на рассмотренном ранее материале и содержат практические вопросы для закрепления полученных знаний. Ответы на практические вопросы приведены в приложении А.

Первые семь глав содержат основные понятия реляционных баз данных и SQL, за ними следуют основы запросов (queries). Запросы — команды, используемые для поиска данных в базах данных; они представляют собой наиболее общий и наиболее сложный аспект SQL. В главах с 8 по 14 техника запросов усложняется. Вводятся различные способы комбинирования запросов и запросы более чем к одной таблице. Другие аспекты SQL: создание таблиц, ввод в них значений, предоставление и закрытие доступа к созданным таблицам — рассмотрены в главах с 15 по 23. Глава 24 показывает, как получить доступ к информации о структуре базы данных. В главе 25 речь идет об использовании SQL в программах, написанных на других языках.

В зависимости от того, как будет использоваться SQL, часть информации, расположенной в конце книги, может не пригодиться. Не все пользователи создают таблицы или вводят в них значения. Эта книга построена таким образом, что каждая следующая глава продолжает предыдущую, но можно свободно пропускать те разделы, которые никогда не придется использовать. Именно по этой причине введение в запросы полностью представлено в начале книги. Запросы — это основа, необходимая для того, чтобы успешно применять большинство других функций SQL.

Во всем множестве примеров, представленных в книге, будет использоваться единый набор таблиц.

Содержимое книги по главам выглядит следующим образом:

- Глава 1 дает понятие реляционной базы данных и концепции первичных ключей (primary keys). В ней также приводятся и поясняются три таблицы, на которых базируется множество представленных в книге примеров.

- Глава 2 ориентирует вас в мире SQL. В ней рассматриваются важные вопросы структуры языка, различные типы данных, распознаваемые SQL, некоторые общие соглашения SQL и терминология.

- Глава 3 учит создавать запросы и знакомит с несколькими приемами по их уточнению. После изучения этой главы вы сможете использовать SQL с практической пользой.

- Глава 4 иллюстрирует, каким образом применяются в SQL два типа стандартных математических операторов, отношения (=, <, >, и т.д.) и булевы операции (AND, OR, NOT).

- Глава 5 вводит ряд операторов, которые используются так же, как операторы отношения, но являются специфичными для SQL. В этой главе даются разъяснения по вопросу потери данных, и определены NULL-значения.

- Глава 6 учит применять операторы, позволяющие выводить данные на основе тех, которые хранятся в таблицах, способом, отличным от простого извлечения. Это дает возможность суммировать значения данных, хранящихся в таблицах.

- Глава 7 поясняет ряд действий, возможных при выводе запроса: выполнение математических операций над данными, включение текста, сортировка.

- Глава 8 показывает, как простой запрос может извлекать информацию более чем из одной таблицы. Этот процесс определяет связь таблиц, включая способы оперирования с данными.

- Глава 9 демонстрирует технику получения ответа на запрос по множеству таблиц, применимую к установлению специальной связи для одной таблицы.

- Глава 10 научит выполнять запрос и использовать его результат в другом запросе.

- Глава 11 расширяет технику, рассмотренную в главе 10, и учит использовать вложенные запросы многократно.

- Глава 12 вводит новый тип специального оператора SQL. EXISTS — оператор, действующий на весь запрос, а не на отдельное простое значение.

- Глава 13 вводит новый тип операторов — ANY, ALL, SOME, которые, подобно оператору EXISTS, действуют на весь запрос.

- Глава 14 вводит команды, позволяющие непосредственно комбинировать результаты множественных запросов способом, отличным от их последовательного выполнения.

- Глава 15 вводит команды, позволяющие определить, какие значения хранятся в базе данных, а также команды вставки, удаления и обновления значений.

- Глава 16 расширяет мощность только что введенных команд. В ней показано, как запросы могут управлять их выполнением.

- Глава 17 учит создавать новую таблицу.

- Глава 18 детально объясняет процесс создания таблиц. Вы узнаете, как предусмотреть отказ от автоматического выполнения некоторого вида изменений.

- Глава 19 исследует логические связи, существующие между данными, на основе совпадения значений.

- Глава 20 рассказывает о представлениях, об "окне", разворачивающем таблицу, отличную от той, что хранится в базе данных.

- Глава 21 касается сложных вопросов изменения значений в представлениях, когда вы реально изменяете соответствующие таблицы. Именно с этим связана здесь необходимость рассмотрения специальных вопросов.

- Глава 22 рассказывает о привилегиях: кто имеет право обращаться с запросами к таблицам, кто имеет право изменять их содержимое, как эти права назначаются пользователям, как пользователи их лишаются и т.д.

- Глава 23 представляет некоторые ранее не рассмотренные важные моменты. Например, мы обсудим те изменения базы данных, которые становятся постоянными, а также выполнение ряда операций в SQL.

- Глава 24 описывает, как SQL поддерживает структурирование баз данных и каким образом осуществляется доступ к ним.

- Глава 25 фокусирует внимание на специальных проблемах и процедурах, связанных с вводом SQL-команд из других языков. Здесь же рассмотрены аспекты языка, специфичные для встроенной формы, например, курсоры и команда FETCH.

В приложениях вы найдете ответы на вопросы (приложение A), описание таблиц, рассматриваемых в качестве примеров (приложение B), детальные сведения о различных типах данных (приложение C), общие элементы, отличные от стандарта (приложение D), руководство по командам SQL (приложение E), взгляд на современный SQL (приложение F).

Соглашения, принятые в этой книге

SQL состоит из инструкций, которые передаются программе, управляющей работой базы данных, предлагая ей выполнить определенные действия. Эти инструкции в общем виде называют предложениями, но мы в большинстве случаев будем использовать термин "команды", чтобы показать, что они имеют область действия.

Термины выделены курсивом в тех местах, где они в первый раз встречаются. В синтаксисе команд курсив используется для того, чтобы показать, что слова имеют дополнительный смысл.

В примерах представлен текст, который следует ввести в программу обработки базы данных, и показан результат для конкретного программного продукта (FirstSQL, программа, работающая с базой данных на IBM PC). Результат, полученный с помощью других программных продуктов, может отличаться от приведенного, но основной результат (данные, полученные из базы данных) не зависит от конкретного программного продукта.

1

Введение в реляционные базы данных

Прежде чем начать использовать SQL, вы должны понять, что такое реляционная база данных. Мы намеренно не будем обсуждать в этой главе SQL, поэтому вы можете пропустить ее, если достаточно хорошо владеете основным понятиями реляционных баз данных. Однако в любом случае следует взглянуть на три таблицы, представленные в конце главы, поскольку именно они используются в большинстве примеров, приведенных в книге. Вы также можете ознакомиться с ними в приложении E. Мы рекомендуем постоянно иметь копию этих таблиц перед глазами.

Что такое реляционная база данных?

Реляционная база данных — это связанная информация, представленная в виде двумерных таблиц. Представьте себе адресную книгу. Она содержит множество строк, каждая из которых соответствует данному индивидууму. Для каждого из них в ней представлены некоторые независимые данные, например, имя, номер телефона, адрес. Представим такую адресную книгу в виде таблицы, содержащей строки и столбцы. Каждая строка (называемая также *записью*) соответствует определенному индивидууму, каждый столбец содержит значения соответствующего типа данных: имя, номер телефона и адрес, — представленных в каждой строке. Адресная книга может выглядеть таким образом:

Name	**Telephone**	**Address**
(Имя)	**(Телефон)**	**(Адрес)**
Gerry Farish	(415)365-8775	127 Primrose Ave., SF
Celia Brock	(707) 874-3553	246 #4 3rd St., Sonoma
Yves Grillet	(762)976-3665	778 Modernas, Barcelona

То, что мы получили, является основой реляционной базы данных, определенной в начале нашего обсуждения двумерной (строки и столбцы) таблицей информации. Однако, реляционная база данных редко состоит из одной таблицы, которая слишком мала по сравнению с базой данных. При создании нескольких таблиц со связанной информацией можно выполнять более сложные и мощные операции над данными. Мощность базы данных заключается, скорее, в связях, которые вы конструируете между частями информации, чем в самих этих частях.

Установление связи между таблицами

Давайте используем пример адресной книги для того, чтобы обсудить базу данных, которую можно реально использовать в деловой жизни. Предположим, что индивидуумы первой таблицы являются пациентами больницы. Дополнительную информацию о них можно хранить в другой таблице. Столбцы второй таблицы могут быть поименованы таким образом: Patient (Пациент), Doctor (Врач), Insurer (Страховка), Balance (Баланс).

Patient (Пациент)	**Doctor (Врач)**	**Insurer (Страховка)**	**Balance (Баланс)**
Farish	Drume	B.C./B.S.	$272.99
Grillet	Halben	None	$44.76
Brock	Halben	Health, Inc.	$9077.47

Можно выполнить множество мощных функций при извлечении информации из этих таблиц в соответствии с заданными критериями, особенно, если критерий включает связанные части информации из различных таблиц. Предположим, Dr. Halben желает получить номера телефонов всех своих пациентов. Для того чтобы извлечь эту информацию, он должен связать таблицу с номерами телефонов пациентов (адресную книгу) с таблицей, определяющей его пациентов. В данном простом примере он может мысленно проделать эту операцию и узнать телефонные номера своих пациентов Grillet и Brock, в действительности же эти таблицы вполне могут быть больше и намного сложнее. Программы, обрабатывающие реляционные базы данных, были созданы для работы с большими и сложными наборами тех данных, которые являются наиболее общими в деловой жизни общества. Даже если база данных больницы содержит десятки или тысячи имен (как это, вероятно, и бывает в реальной жизни), единственная команда SQL предоставит доктору Halben необходимую информацию практически мгновенно.

Порядок строк произволен

Для обеспечения максимальной гибкости при работе с данными строки таблицы, по определению, никак не упорядочены. Этот аспект отличает базу данных от адресной книги. Строки в адресной книге обычно упорядочены по алфавиту. Одно из мощных средств, предоставляемых реляционными системами баз данных, состоит в том, что пользователи могут упорядочивать информацию по своему желанию.

Рассмотрим вторую таблицу. Содержащуюся в ней информацию иногда удобно рассматривать упорядоченной по имени, иногда — в порядке возрастания или убывания баланса (Balance), а иногда — сгруппированной по доктору. Внушительное множество возможных порядков строк помешало бы пользователю проявить гибкость в работе с данными, поэтому строки предполагаются неупорядоченными. Именно по этой причине вы не можете просто сказать: "Меня интересует пятая строка таблицы". Независимо от порядка включения данных или какого-либо другого критерия, этой пятой строки не существует по определению. Итак, строки таблицы предполагаются расположенными в произвольном порядке.

Идентификация строк (первичный ключ)

По этой и ряду других причин, необходимо иметь столбец таблицы, который однозначно идентифицирует каждую строку. Обычно этот столбец содержит номер, например, приписанный каждому пациенту. Конечно, можно использовать для идентификации строк имя пациента, но ведь может случиться так, что имеется не-

сколько пациентов с именем Mary Smith. В подобном случае нет простого способа их различить. Именно по этой причине обычно используются номера. Такой уникальный столбец (или их группа), используемый для идентификации каждой строки и обеспечивающий различимость всех строк, называется *первичным ключом таблицы (primary key of the table)*.

Первичный ключ таблицы — жизненно важное понятие структуры базы данных. Он является сердцем системы данных: для того чтобы найти определенную строку в таблице, укажите значение ее первичного ключа. Кроме того, он обеспечивает целостность данных. Если первичный ключ должным образом используется и поддерживается, вы будете твердо уверены в том, что ни одна строка таблицы не является пустой и что каждая из них отлична от остальных. Ключи мы рассмотрим позже, после обсуждения ссылочной целостности (referential integrity) в главе 19.

Столбцы поименованы и пронумерованы

В отличие от строк, столбцы таблицы (также называемые *полями (fields)* упорядочены и поименованы. Следовательно, в нашей таблице, соответствующей адресной книге, можно сослаться на столбец "Address" как на "столбец номер три". Естественно, это означает, что каждый столбец данной таблицы должен иметь имя, отличное от других имен, для того, чтобы не возникло путаницы. Лучше всего, когда имена определяют содержимое поля. В этой книге мы будем использовать аббревиатуру для именования столбцов в простых таблицах, например: *cname* — для имени покупателя (customer name), *odate* — для даты поступления (order date). Предположим также, что таблица содержит единственный цифровой столбец, используемый как первичный ключ. В следующем разделе детально объясняются таблицы, используемые в качестве примера и их ключи.

Пример базы данных

Таблицы 1.1, 1.2, 1.3 образуют реляционную базу данных, которая достаточно мала для того, чтобы можно было понять ее смысл, но и достаточно сложна для того, чтобы иллюстрировать на ее примере важные понятия и практические выводы, связанные с применением SQL. Эти же таблицы приведены в приложении E. Поскольку в этой книге они будут использоваться для иллюстрации различных черт SQL, мы рекомендуем скопировать их и постоянно иметь перед глазами. Можно заметить, что первый столбец в каждой таблице содержит номера, не повторяющиеся от строки к строке в пределах таблицы. Как вы, наверное, догадались, это первичные ключи таблицы. Некоторые из этих номеров появляются также в столбцах других таблиц (в этом нет ничего предосудительного), что указывает на связь между строками, использующими конкретное значение первичного ключа, и той строкой, в которой это значение применяется непосредственно в первичном ключе.

Таблица 1.1. Salespeople (Продавцы)

SNUM	SNAME	CITY	COMM
1001	Peel	London	.12
1002	Serres	San Jose	.13
1004	Motika	London	.11
1007	Rifkin	Barcelona	.15
1003	Axelrod	New York	.10

Таблица 1.2. Customers (Покупатели)

CNUM	CNAME	CITY	RATING	SNUM
2001	Hoffman	London	100	1001
2002	Giovanni	Rome	200	1003
2003	Liu	San Jose	200	1002
2004	Grass	Berlin	300	1002
2006	Clemens	London	100	1001
2008	Cisneros	San Jose	300	1007
2007	Pereira	Rome	100	1004

Таблица 1.3. Orders (Заказы)

ONUM	AMT	ODATE	CNUM	SNUM
3001	18.69	10/03/1990	2008	1007
3003	767.19	10/03/1990	2001	1001
3002	1900.10	10/03/1990	2007	1004
3005	5160.45	10/03/1990	2003	1002
3006	1098.16	10/03/1990	2008	1007
3009	1713.23	10/04/1990	2002	1003
3007	75.75	10/04/1990	2004	1002
3008	4723.00	10/05/1990	2006	1001
3010	1309.95	10/06/1990	2004	1002
3011	9891.88	10/06/1990	2006	1001

Например, поле snum в таблице Customers определяет, каким продавцом (salespeople) обслуживается конкретный покупатель (customer). Номер поля snum ус-

танавливает связь с таблицей Salespeople, которая дает информацию об этом продавце (salespeople). Очевидно, что продавец, который обслуживает данного покупателя, существует, т.е. значение поля snum в таблице Customers присутствует также и в таблице Salespeople. В этом случае мы говорим, что система находится в состоянии ссылочной целостности (referential integrity). Это понятие более подробно и формально объясняется в главе 19.

Сами по себе таблицы предназначены для описания реальных ситуаций в деловой жизни, когда можно использовать SQL для ведения дел, связанных с продавцами, их покупателями и заказами. Давайте зафиксируем состояние этих трех таблиц в какой-либо момент времени и уточним назначение каждого из полей таблицы.

Перед вами объяснение столбцов таблицы 1.1:

ПОЛЕ	**СОДЕРЖИМОЕ**
snum	Уникальный номер, приписанный каждому продавцу ("номер служащего")
sname	Имя продавца
city	Место расположения продавца
comm	Вознаграждение (комиссионные) продавца в форме с десятичной точкой

Таблица 1.2 содержит следующие столбцы:

ПОЛЕ	**СОДЕРЖИМОЕ**
cnum	Уникальный номер, присвоенный покупателю
cname	Имя покупателя
city	Место расположения покупателя
rating	Цифровой код, определяющий уровень предпочтения данного покупателя. Чем больше число, тем больше предпочтение
snum	Номер продавца, назначенного данному покупателю (из таблицы Salesperson)

И, наконец, столбцы таблицы 1.3:

ПОЛЕ	СОДЕРЖИМОЕ
onum	Уникальный номер, присвоенный данной покупке
amt	Количество
odate	Дата покупки
cnum	Номер покупателя, сделавшего покупку (из таблицы Customers)
snum	Номер продавца, обслужившего покупателя (из таблицы Salespeople)

Итоги

Итак, теперь вы знаете, чем является реляционная база данных. Вы также познакомились с некоторыми фундаментальными принципами структурирования таблиц, узнали, как работают строки и столбцы, как с помощью первичного ключа можно отличить одну строку таблицы от другой, и, наконец, как столбцы могут ссылаться на значения других столбцов. Вы узнали, что понятие "запись" является синонимом понятия "строка" и что понятие "поле" является синонимом понятия "столбец". Мы тоже будем использовать оба термина при обсуждении SQL в качестве синонимов.

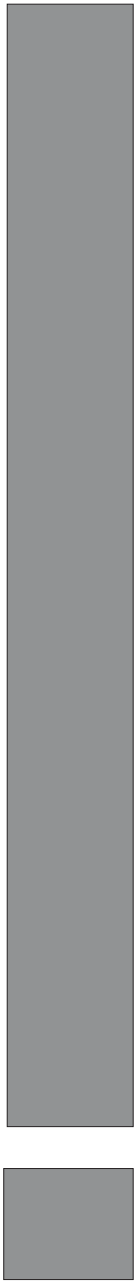
Вы уже знакомы с простыми таблицами. При всей своей краткости и простоте они вполне пригодны для демонстрации наиболее важных черт языка, в чем вы позже сами убедитесь. Иногда мы будем вводить другие таблицы или рассматривать другие данные в одной из этих таблиц для того, чтобы показать некоторые дополнительные возможности их применения.

Теперь мы готовы к непосредственному погружению в SQL. Следующая глава, к которой вам время от времени придется возвращаться, дает общее представление о языке и ориентирует вас в изложенном в книге материале.

Работаем на SQL

1. Какое поле в таблице Customers является первичным ключом?
2. Дайте объяснение столбцу с номером 4 в таблице Customers?
3. Как иначе называются строка и столбец?
4. Почему нельзя попросить показать вам первые пять строк таблицы?

(Ответы см. в приложении А.)



2

Введение в SQL

В этой главе речь пойдет о структуре языка SQL, о некоторых общих вопросах, касающихся типов данных, которые могут содержаться в таблицах, а также о неясностях, существующих в SQL. Здесь же уточняется контекст специфической информации, которая будет дана в последующих главах. Вы можете войти в мир SQL, не упрощая его, и легко вернуться к нужному месту, если возникнут вопросы, благодаря тому, что этот материал расположен в начале книги.

Как работает SQL?

SQL — это язык, ориентированный специально на реляционные базы данных. Он позволяет исключить большую работу, выполняемую при использовании языка программирования общего назначения. Для создания реляционной базы данных, например на языке C, пришлось бы начать с определения объекта, называемого таблицей, который может иметь произвольное число строк, а затем создавать процедуры для ввода значений в таблицу и для поиска в ней данных. Для нахождения каких-то конкретных строк пришлось бы выполнить последовательность действий, например:

1. Посмотреть очередную строку таблицы.
2. Оттестировать ее и убедиться, что это та строка, которая Вас интересует.
3. Запомнить ее до тех пор, пока не будет просмотрена вся таблица.
4. Определить, есть ли в таблице еще строки.
5. Если в таблице еще есть строки (просмотрены не все строки), то вернуться к шагу 1.
6. Если в таблице больше нет строк (просмотрены все строки таблицы), вывести все значения, полученные на третьем этапе.

SQL освобождает от подобной работы. Команды SQL могут выполняться над целой группой таблиц, как над единственным объектом, а также могут оперировать любым количеством информации, которая извлекается или выводится из них как из единого целого.

Как осуществляется связь с ANSI-таблицей?

Стандарт SQL определен ANSI (American National Standards Institute — Американским национальным институтом стандартов). SQL не является изобретением ANSI, он — продукт исследований фирмы IBM. Однако другие компании тоже внесли свою лепту в развитие SQL; по крайней мере, компания Oracle превзошла IBM в создании популярного рыночного программного SQL-продукта.

После того, как на рынке появилось несколько конкурирующих SQL-продуктов, ANSI определила стандарт, которому все они должны удовлетворять. Однако вве-

дение стандарта post factum порождает ряд проблем. Результирующий стандарт SQL в некотором смысле ограничен: то, что определено ANSI, не всегда является наиболее полезным с точки зрения практического применения, поэтому создатели SQL-продуктов стараются разрабатывать их таким образом, чтобы они соответствовали стандарту ANSI, но не были бы слишком жестко ограничены его требованиями. Программные продукты, выполняющие обработку баз данных (системы управления базами данных — СУБД), обычно придают ANSI SQL дополнительные характерные черты и часто снимают большинство существенных практических ограничений, присущих стандарту. Поэтому наиболее общие отклонения от ANSI тоже следует проанализировать. Рассмотреть каждое отдельное исключение и отклонение от стандарта невозможно, однако полезные идеи, как правило, копируются и применяются одинаково в различных программных продуктах, даже если они не специфицированы ANSI. ANSI — это своего рода минимальный стандарт; можно делать гораздо больше, чем в нем определено, но, выполняя стандартную задачу, нужно обеспечить предусмотренные данным стандартом результаты.

Интерактивная версия встроенного SQL

Существуют два SQL: интерактивный и встроенный. В основном эти две формы SQL работают одинаково, но используются по-разному.

Интерактивный SQL применяется для выполнения действий непосредственно в базе данных с целью получить результат, который используется человеком. При применении этой формы SQL вводится команда, она выполняется, после чего можно немедленно увидеть выходные данные (если таковые есть).

Встроенный SQL состоит из команд SQL, включенных в программы, которые в большинстве случаев написаны на каком-то другом языке программирования (например, Cobol или Pascal). Такое включение может сделать программу более мощной и эффективной. Однако, несовместимость этих языков программирования со структурой SQL и присущим ему стилем управления данными требует внесения ряда расширений в интерактивный SQL. Выходные данные команд SQL во встроенном SQL "заносятся" в переменные или параметры, используемые программой, в которую включены предложения SQL.

В этой книге представлена интерактивная форма SQL, что позволит обсуждать команды и их действие, не обращая внимания на то, как они взаимодействуют с другими языками. Именно интерактивный SQL наиболее полезен для непрограммистов. Все, что характерно для интерактивного SQL, справедливо и для его встроенной формы. Изменения, которые следует выполнить в связи со встроенной формой, рассматриваются в последней главе этой книги.

Подразделы SQL

Как в интерактивном, так и во встроенном SQL имеется множество секций или подразделов. В процессе освоения SQL придется придерживаться данной терминологии, однако неудачным является то, что эти термины не используются всегда и во всех реализациях SQL. Им придается особое значение в ANSI, и они полезны на концептуальном уровне, но во многих SQL-продуктах они практически не выделены, и поэтому стали функциональными категориями SQL-команд.

Язык определения данных (Data Definition Language, DDL; в ANSI он называется также языком определения схемы (Schema Definition Language)) состоит из тех команд, которые создают объекты (таблицы, индексы, представления) в базе данных. Язык манипулирования данными (Data Manipulation Language, DML) — это множество команд, определяющих, какие данные представлены в таблицах в любой момент времени. Язык управления данными (Data Control Language, DCL) состоит из предложений, определяющих, может ли пользователь выполнить отдельное действие. Согласно ANSI, DCL является частью DDL. Важно не путать эти названия. Речь идет не о различных языках как таковых, а о разделах команд SQL, сгруппированных в соответствии с их функциональным назначением.

Различные типы данных

Не все типы значений, содержащиеся в полях таблицы, логически одинаковы. Наиболее очевидны различия между числами и текстом. Невозможно расположить числа в алфавитном порядке или извлечь одно имя из другого. Поскольку системы реляционных баз данных основаны на связях между частями информации, различные типы данных должны явно отличаться друг от друга, чтобы можно было применить подходящие способы их обработки и сравнения.

В SQL каждому полю приписывается "тип данных" (data type), который определяет, какого рода значения могут содержаться в поле. Все значения для данного поля должны быть одного типа. В таблице Customers, например, поля cname и city являются строками текста, тогда как поля rating, snum, cnum — числовые. Именно по этой причине невозможно занести значения "Highest" или "None" в поле rating, имеющее числовой тип. Это удачное ограничение, поскольку оно накладывает некоторую структуру на конкретные данные. Операцию сравнения, которая выполняется для одних строк и не выполняется для других, невозможно произвести, если значения поля имеют смешанный тип данных.

Определение этих типов данных является той областью, в которой многие коммерческие СУБД и официальный стандарт SQL имеют существенные различия. Стандарт ANSI SQL распознает только текстовый и числовой типы, тогда как многие коммерческие СУБД используют и другие специальные типы данных. Заметим, что типы DATE (дата) и TIME (время) почти de-facto являются стандартными (хотя конкретные их форматы отличаются). Некоторые СУБД поддерживают такие типы данных как MONEY (деньги) и BINARY (двоичный). (BINARY — это специальное числовое

представление, используемое компьютером. Вся информация в компьютере представлена двоичными числами, затем она преобразуется в другие системы — так ее легче использовать и понимать.)

ANSI определяет несколько различных типов числовых значений. Типы данных ANSI полностью перечислены в приложении B. Сложность числовых типов ANSI объясняется, по крайне мере частично, попыткой поддержать совместимость вложенного SQL с множеством других языков.

Два типа данных ANSI, INTEGER и DECIMAL (для которых можно использовать аббревиатуру INT и DEC соответственно), адекватны и теоретическим целям, и множеству практических приложений в деловой жизни. INTEGER отличается от DECIMAL тем, что запрещает использовать цифры справа от десятичной точки, а также саму десятичную точку.

Типом данных для текста является CHAR (CHARACTER), который относится к строке текста. Поле типа CHAR имеет фиксированную длину, равную максимальному числу букв, которые можно ввести в это поле. Большинство реализаций SQL имеет нестандартный тип, названный VARCHAR, — это текстовая строка любой длины вплоть до максимума, определяемого конкретной реализацией SQL. Значения CHAR и VARCHAR заключаются в одиночные кавычки, как, например, 'текст'. Различие между ними состоит в том, что для типа CHAR отводится участок памяти, достаточный для хранения строки максимальной длины, а для VARCHAR память выделяется по мере необходимости.

Символьные типы состоят из всех символов, которые можно ввести с клавиатуры, в том числе и цифр. Однако, число 1 не есть то же самое, что символ '1'. Символ '1' это совсем другая часть печатного текста, которая не распознается компьютером как числовое значение 1. 1 + 1 = 2, но '1' + '1' не равно '2'. Значения типа CHARACTER хранятся в компьютере как двоичные значения, но для пользователя представляются в виде печатного текста. Преобразование выполняется в соответствии с форматом, определяемым той системой, которой вы пользуетесь. Это может быть формат одного из двух стандартных типов (возможно, с расширениями), которые применяются в компьютерных системах : ASCII (используется во всех персональных и большинстве малых компьютеров) и EBCDIC (используется для больших компьютеров). Определенные операции, такие как упорядочение значений поля по алфавиту, зависят от формата. Значения этих двух форматов будут рассмотрены в главе 4.

Тип DATE будет применяться в соответствии с требованиями рынка, а не ANSI. В реализациях SQL, не распознающих тип DATE, можно объявить дату символьным или числовым полем, но это затруднит выполнение множества операций. Следует ознакомиться с документацией по программному обеспечению SQL-системы, чтобы точно определить, какие типы данных она поддерживает.

Кто такой "пользователь"?

SQL устанавливается, как правило, в компьютерных системах, имеющих не одного, а многих пользователей, которых нужно уметь различать (у семейного PC может быть любое число пользователей, но обычно не существует способа отличить их друг от друга). В типичной ситуации каждый пользователь такой системы имеет

код авторизации, который идентифицирует его или ее (в терминологии имеются различия). В начале сеанса связи с компьютером пользователь *регистрируется* в системе, сообщая компьютеру, какой именно пользователь, идентифицированный кодом авторизации ID, находится на связи. Что касается компьютера, то любое число пользователей, имеющих один и тот же ID, является для него одним пользователем; напротив, один человек может восприниматься как множество пользователей, если он (обычно в различные моменты времени) использует различные коды авторизации ID.

SQL придерживается этого правила. В большинстве SQL-систем действия приписываются определенному ID, который обычно соответствует определенному пользователю. Таблица (или другой объект) принадлежит тому пользователю, который имеет на нее (или на этот объект) полномочия. Пользователь может иметь или не иметь привилегию работы с объектами, которые ему не принадлежат. В главе 22 специально обсуждаются привилегии, пока же предположим, что любой пользователь имеет привилегию выполнять любые необходимые ему действия.

Специальное значение USER может использоваться как аргумент в команде. Он обозначает авторизационный ID пользователя, дающего команду.

Соглашения и терминология

Ключевые слова это слова, имеющие специальное значение в SQL. Они являются инструкциями, а не текстом или именами объектов. Ключевые слова будут выделяться заглавными буквами. Следует быть внимательнее и не путать ключевые слова с терминами. SQL имеет определенный набор специальных терминов, которые применяются для его описания. Среди них есть такие слова как запрос, предложение, предикат. Они важны для описания и понимания языка, но для самого SQL ничего не значат.

Команды (*commands*) или *сообщения* (*statements*) — это инструкции, которые даются базе данных SQL. Команды состоят из одной или более логически различных частей, называемых *предложениями* (*фразами*, *clauses*). Предложения начинаются с ключевого слова, по которому они обычно и называются, и состоят из ключевых слов и аргументов. Примерами предложений являются: "FROM Salespeople" и "WHERE city = 'London'". *Аргументы* заканчивают предложение или модифицируют его смысл. В приведенных примерах "Salespeople" является аргументом, а FROM - ключевым словом предложения FROM. Также "city = 'London'" является аргументом предложения WHERE. Объекты — это структуры в базе данных, которые имеют имена и хранятся в памяти. Они включают базовые таблицы, представления (то есть два вида таблиц) и индексы.

Объяснение того, как формулируются команды, будет осуществляться в основном на примерах. Однако существует более формальный метод описания команд с использованием стандартных соглашений, который иногда применяется в следующих главах. Упомянутые соглашения полезно знать в случае столкновения с ними в другой документации по SQL. Квадратные скобки ([ ]) выделяют те части, которые можно опустить, круглые скобки (...) показывают, что предшествующее им можно повторить

любое число раз. Слова, заключенные в угловые скобки (< >), — специальные термины, которые объясняются по мере того, как вводятся.

Итоги

Эта глава охватывает большое количество основной информации, дающей общее представление об SQL. Вы узнали, как он структурирован, как используется, как в нем выражаются данные, как и кем он определяется (и какие противоречия при этом возникают), а также некоторые соглашения и терминологию, используемые для описания.

В следующей главе подробно объясняются формирование и действие команд. Вы познакомитесь с командой, позволяющей извлекать информацию из таблиц и являющейся одной из наиболее часто применяемых в SQL. Вы сможете вывести сами определенную информацию из базы данных.

Работаем на SQL

1. Каковы основные различия между типами данных в SQL?
2. Есть ли в ANSI тип данных DATE?
3. Какой подраздел SQL используется для ввода значений в таблицы?
4. Что такое ключевое слово?

(Ответы даны в приложении А.)