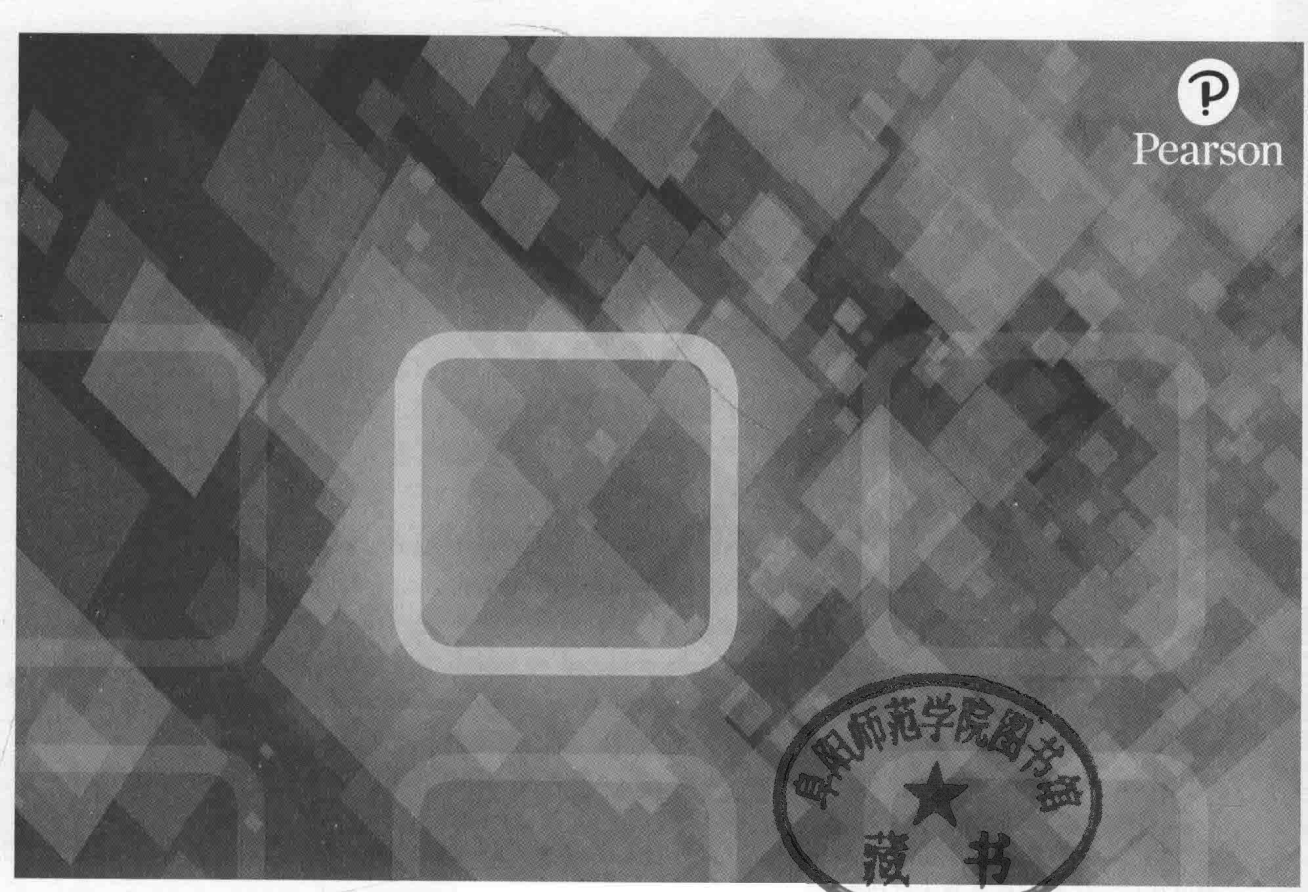
The Language of SQL SECOND EDITION

SQL 初学者指南（第2版）

[美] Larry Rockoff 著
李强 译

中国工信出版集团
人民邮电出版社
POSTS & TELECOM PRESS



SQL初学者指南（第2版）

[美] Larry Rockoff 著
李强 译

人民邮电出版社
北京

图书在版编目（CIP）数据

SQL初学者指南 : 第2版 / (美) 洛克夫(Larry Rockoff) 著 ; 李强译. -- 北京 : 人民邮电出版社, 2017.4
ISBN 978-7-115-44865-1

Ⅰ. ①S… Ⅱ. ①洛… ②李… Ⅲ. ①关系数据库系统—指南 Ⅳ. ①TP311.138-62

中国版本图书馆CIP数据核字(2017)第030787号

◆ 著　[美] Larry Rockoff
译　李　强
责任编辑　陈冀康
责任印制　焦志炜

◆ 人民邮电出版社出版发行　北京市丰台区成寿寺路 11 号
邮编　100164　电子邮件　315@ptpress.com.cn
网址　http://www.ptpress.com.cn
北京天宇星印刷厂印刷

◆ 开本：800×1000　1/16
印张：13.25
字数：271 千字　2017 年 4 月第 1 版
印数：1－3 000 册　2017 年 4 月北京第 1 次印刷

著作权合同登记号　图字：01-2016-8606 号

定价：49.00 元

读者服务热线：(010)81055410　印装质量热线：(010)81055316
反盗版热线：(010)81055315
广告经营许可证：京东工商广字第 8052 号

目录

第1章
关系型数据库和SQL

正如前言中所提到的，在与关系型数据库中的数据进行交互的时候，SQL是使用最广泛的软件工具。在这方面，SQL利用了自身的语言和逻辑两方面的要素。作为一种语言，SQL的独特语法用到了很多的英语单词，诸如WHERE、FROM和HAVING。作为一种逻辑表达，它指定了在关系型数据库中检索和修改数据的细节。

考虑到了这两方面因素，我们在本书中介绍SQL的各个方面的时候，尝试强调语言和逻辑这两部分。在所有语言中，无论它们是计算机语言还是口语，我们都需要学习和记忆实际的单词。因此，我们以逻辑顺序来介绍SQL语句中出现的不同的关键字，并且每次介绍一个关键字。当学习完每一章之后，你就能够在之前的“单词量”的基础上，学习新的关键字以及使用它与数据库交互的激动人心的潜力。

除了单词本身，还有一些逻辑有待考虑。SQL语句使用的单词有独特的逻辑含义和意图。SQL的逻辑和语言一样重要。和所有的计算机语言一样，SQL经常使用不止一种方法来指定任何所需的目标。这些方法之间的细微差别，可能包括语言和逻辑两个方面。

让我们开始学习这门语言！在熟悉了SQL的语法之后，你可能会发现，SQL命令的思维方式和英语语句很类似，并且也能表达某种含义。

例如，对比下面这句话：

```
I would like a hamburger and fries
from your value menu,
and make it to go.
```

和这条SQL语句：

```
Select city, state
from Customers
order by state
```

这条SQL语句表示我们想要从数据库的Customers表中获取city和state字段，并且希望结果按照state来排序。我们稍后详细介绍这条语句的具体细节。

在这两个示例中，我们指定了想要的东西（hamburger/fries或city/state），从哪里获取（value菜单或Customers表），以及一些额外的指令（尽快去做或将结果按照state来排序）。

但是在开始之前，让我们先解决一个小问题：SQL 这个单词是怎么念的？事实上有两种方法。一种方法是直接按照单个的字母来发音，就像“S-Q-L”。另一种选择，也是本书作者的首选，把这个单词读作“sequel”。这样少一个音节，也更容易读出来。然而，这个问题没有一个标准的答案。具体怎么念，实际上是个人的喜好。

大多数人都认同，字母 S-Q-L 的含义表示结构化查询语言（Structured Query Language）。然而，即使在这一点上，也并没有达成完全的共识。有些人认为 SQL 根本不表示任何含义，因为它是从一种叫做 sequel 的、古老的 IBM 语言派生而来，而这门古老的语言实际上并不代表结构化查询语言。

1.1　SQL 是什么

那么 SQL 到底是什么呢？简而言之，SQL 就是为了维护和使用关系型数据库中的数据而使用的一种标准的计算机语言。简单来说，SQL 就是能让用户和关系型数据库进行交互的一种语言。SQL 语言有很长的发展历史，很多组织都对它的发展做出了贡献，它最早的历史可以追溯到 20 世纪 70 年代。1986 年，美国国家标准局（American National Standards Institute，ANSI）发布了该语言的第一套标准，从那时起，它经历过多次的修订。

一般来讲，SQL 语言有 3 个主要的组成部分。第 1 个部分叫做数据操纵语言（Data Manipulation Language，DML）。SQL 语言的这个模块让我们可以检索、修改、增加或删除数据库中的数据。第 2 个部分叫做数据定义语言（Data Definition Language，DDL）。DDL 使得我们能够创建和修改数据库本身。例如，DDL 提供了 ALTER 语句，它让我们可以修改数据库中的表的设计。第 3 个部分是数据控制语言（Data Control Language，DCL），用于维护数据库的安全。

许多主要的软件厂商，像 Microsoft 和 Oracle，为了其各自的目的，都会修改这个标准，并且对该语言添加大量的扩展和修改。尽管每个厂商对于 SQL 都有自己独特的解释，但是仍然会有一个底层的基础语言，而它对于所有厂商几乎都是一致的。这个基础语言，也正是本书所要介绍的内容。

作为一种计算机语言，和你可能熟悉的其他语言（如 Visual Basic 或 C++）相比，SQL 并不相同。其他语言本质上往往趋向于过程化。这就意味着，它们允许你指定特定的过程来完成想要实现的任务。SQL 更趋向于一种声明式语言（Declarative Language）。在 SQL 中，经常用一条单独的语句来声明预期的目标。SQL 的结构之所以如此简单，是因为它只关注关系型数据库，而不是整个计算机系统。

关于 SQL 语言，还有一点需要澄清，人们经常会把 SQL 语言和具体的 SQL 数据库搞混。有很多的软件公司销售数据库管理系统（Database Management System，DBMS）。通常，这些类型的软件包中的数据库称为 SQL 数据库，因为 SQL 语言是管理和访问这

些数据库中的数据的主要方法。一些厂商甚至把 SQL 作为其数据库名称的一部分。例如，Microsoft 把它最新的数据库叫做 SQL Server 2016。但实际上，更准确地讲，SQL 是一种语言，而不是一个数据库。本书的重点是介绍 SQL 的语言，而不是介绍任何一种特定的数据库。

1.2 Microsoft SQL Server、MySQL 和 Oracle

尽管我们的目标是介绍 SQL 的核心语言，因为它适用于所有的实现，但是，我也会提供 SQL 语法的一些具体示例。由于各个厂商的语法各异，我决定重点关注如下这 3 种数据库所使用的 SQL 语法：

- Microsoft SQL Server；
- Oracle；
- MySQL。

大多数情况下，这些数据库有着相同的语法。然而，偶尔也会有所不同。如果这 3 种数据库之间有任何的差异，本书的正文中将会采用 Microsoft SQL Server 的语法，我会像下面这样，专门指出 MySQL 或 Oracle 在语法上的不同之处。

> **数据库的差异**
> *当我要介绍 Oracle 数据库或 MySQL 数据库的不同语法时，就会以“数据库的差异”这样的版块给出。正文中则会给出 Microsoft SQL Server 的语法。*

Microsoft SQL Server 有好几个可用的版本。最新的版本叫做 Microsoft SQL Server 2016；既有基础的 Express 版，又有功能齐全的企业（Enterprise）版。尽管 Express 版是免费的，但是它仍然有着丰富的功能，可以用来进行完整的数据库开发。企业版包括许多高级的数据库管理功能以及高级的商务智能组件。

尽管 MySQL 属于 Oracle 公司，但它是一款开源的数据库，这意味着没有一家独立的机构控制其开发。除了 Windows 外，MySQL 还可以在许多平台上运行，诸如 Mac OS X 和 Linux。MySQL 提供了社区版（Community Edition）供免费下载。其最新版本是 MySQL 5.7。

Oracle 数据库也有多个可用的版本。最新的版本是 Oracle Database 12c。Oracle 数据库的免费版本叫做 Express 版。

作为初学者，请根据自己的选择去下载数据库，以便能够进行尝试，这么做有时候是很有用的。但是，本书并不需要你这么做。本书的编写方法是，允许你通过只阅读正文来学习 SQL。在正文中，我会提供足够的数据，你无需下载软件或亲自输入语句，也能理解各种 SQL 语句的结果。

尽管如此，如果你想要下载这些数据库的免费版本，本书的附录 A 到附录 C 针对如何下载给出了一些介绍和建议。附录 A 针对如何开始使用 Microsoft SQL Server 给出了详尽的说明，包括如何安装软件以及执行 SQL 命令的详细介绍。附录 B 介绍的是 MySQL，而附录 C 介绍的是 Oracle。

正如前言所提到的，本书的配套网站提供了本书在 3 种数据库中所用到的、所有 SQL 语句的辅助材料。然而，你很可能会发现根本不需要下载或阅读配套网站上的补充材料。本书中所有的示例，都是一看便知的，不需要为了理解这些内容而做任何事情。但是，如果你愿意去下载的话，那就利用好这些额外材料吧！

此外应该注意的是，除了 Microsoft SQL Server、Oracle 和 MySQL 以外，还有一些其他的比较流行的关系型数据库值得了解。例如：

- IMB 的 DB2；
- IBM 的 Informix；
- Sybase 的 SQL Anywhere；
- 开源数据库 PostgreSQL；
- Microsoft 的 Microsoft Access。

在这些数据库中，Microsoft Access 有一些特别，它有一个图形化元素。其实，Access 是关系型数据库的一个图形化界面。换句话讲，Access 允许我们完全通过图形化的方法，来为关系型数据库创建一个查询。对于初学者来讲，Access 最有用的一点，就是可以用可视化的方法很容易地创建一个查询，然后切换到 SQL 视图去查看刚刚创建的 SQL 语句。Access 的另一个不同之处在于，它是一个桌面数据库。因此，我们不仅可以使用 Access 来创建一个完全以单个的文件形式保存在计算机上的数据库，而且还可以连接到用其他工具（诸如 Microsoft SQL Server）创建的数据库。

1.3　关系型数据库

介绍完了这些预备知识，我们来了解一下关系型数据库的基础知识以及它们是如何工作的。关系型数据库就是一个数据集合，它保存了任意多个表。在常见的用法中，术语“关系（relational）”用来表示各个表以某种形式彼此相互关联。然而，更准确地讲，关系指的是数学关系理论，并且和一些逻辑属性相关，而这些逻辑属性负责管理表之间相关的方式。

例如，我们来看数据库的一个简单示例，它只有两个表：Customers 表和 Orders 表。Customers 表为每位下订单的客户保存一条记录。Orders 表针对每个订单保存一条记录。每个表都可以包含任意多个字段，字段用来存储与每条记录相关的不同属性。例如，Customers 表可以保存诸如 FirstName 和 LastName 这样的字段。

这时，可视化一些表和表中所包含的数据是很有用的。通常习惯是，把表显示为由行和列组成的一个表格。每一行表示表中的一条记录，每一列表示表中的一个字段。表的顶行通常是字段名。剩余的其他行则显示实际的数据。

在 SQL 术语中，记录（record）和字段（field）实际上就称为行（row）和列（column），这和视觉上的展示是对应的。因此，今后我们使用术语“行”和“列”来说明关系型数据库中表的设计，而不再使用记录和字段。

我们来看关系型数据库中一个可能的、最简单的示例。在这个数据库中，只有两个表，分别是 Customers 表和 Orders 表。这两个表看上去如下所示。

Customers 表

CustomerID	FirstName	LastName
1	Bob	Davis
2	Natalie	Lopez
3	Connie	King

Orders 表：

OrderID	CustomerID	OrderAmount
1	1	50.00
2	1	60.00
3	2	33.50
4	3	20.00

在这个示例中，Customers 表包含了 3 个列：CustomerID、FirstName 和 LastName。目前，表中有 3 行，分别表示 Bob Davis、Natalie Lopez 和 Connie King。每一行表示一个不同的客户，每一列表示该客户的一段不同的信息。与之类似，Orders 表有 4 行和 3 列。这表示数据库中有 4 笔订单，每笔订单有 3 种属性。

当然，这个示例非常简单，并且只是提示了哪些数据类型可以存储到一个真实的数据库中。例如，Customers 表通常会包含描述客户的其他属性的许多附加的列，诸如 city、state、zip 和 phone number。同理，Orders 表一般也会有一些描述订单的其他属性的列，诸如 order date、sales tax 以及接受该订单的 salesperson。

1.4 主键和外键

请留意每个表的第 1 列：即 Customers 表中的 CustomerID 和 Orders 表中的 OrderID。这些列通常称为主键（primary key）。主键之所以有用和有必要，有两个原因。首先，它们使你能够唯一地标识表中一个单独的行。例如，如果想要查找 Bob Davis 这一行，我们可以只使用 CustomerID 列来获取数据。主键还确保了唯一性。当指定 CustomerID 列作为主键时，就保证了表中的该列针对每一行都拥有一个唯一的值。即使在数据库中有两个不同的人都叫做 Bob Davis，这两行的 CustomersID 列的值也会不同。

在这个示例中，主键列的值没有任何特殊含义。在 Customers 表中，CustomerID 列在表的 3 行中的值分别为 1、2 和 3。我们会经常以这样的一种方式来设计数据库的表：当表中增加新的行时，主键列会自动生成顺序的编号。通常，我们把这种设计特性叫做自增型（auto-increment）。

使用主键的第 2 个原因是，可以很容易地把一个表和另一个表进行关联。在这个例子中，Orders 表中的 CustomerID 列，指向了 Customers 表中对应的一行。查看一下 Orders 表的第 4 行，会发现其 CustomerID 列的值是 3。这就意味着，CustomerID 为 3 的客户下了这个订单，这位客户名为 Connie King。在表之间使用共同的列，这是关系型数据库中的一项基本的设计要素。

除了可以指向 Customers 表，还可以把 Orders 表中的 CustomerID 列指定为外键（foreign key）。我会在第 18 章中详细介绍外键，在这里只需要知道：定义外键是为了要确保这一列有一个有效的值。例如，我们希望 Orders 表中的 CustomerID 列的值，必须是 Customer 表中真正存在的一个 CustomerID 值。指定一列作为外键，就可以实现这种限制。

1.5　数据类型

主键和外键为数据库表添加了结构。它们确保了数据库中所有的表都是可访问的并且表之间有正确的关联。表中的每一列的另一个重要属性是其数据类型。

数据类型是定义一个列所能包含数据的类型的一种方法。要为每个表中的每一列都指定一个数据类型。遗憾的是，各种关系型数据库所允许的数据类型以及它们所代表的含义，有很大的不同。例如，Microsoft SQL Server、MySQL 和 Oracle，各自都有超过 30 种不同的、可用的数据类型。

即使只有 3 种数据库，我们也不可能去介绍每种可用的数据类型的细节及细微差别。但是，我所要做的，是通过讨论大部分数据库中常用的数据类型的主要类别，来概括这种情况。一旦了解了这些类别中的重要的数据类型，当遇到其他可能的数据类型时，也都可以迎刃而解。一般来讲，有 3 种重要的数据类型：数字（Numeric）、字符（Character）以及日期/时间（Date/Time）。

数字数据类型有很多种，包括位（bit）、整数（integer）、浮点数（decimal）和实数（real number）。bit 是数字类型，它只允许有两个值，0 和 1。bit 也经常用来定义只有 true 和 false 值的一个属性。integer 是没有小数点的数字。decimal 可以包含小数点。与 bit、integer 和 decimal 不同，实数的精确值只能是在内部近似地定义。所有数字类型的一个共同的显著特征，就是它们都能用于算术运算中。如下是 Microsoft SQL Server、MySQL 和 Oracle 中的数字类型的一些典型示例。

通用说明	Microsoft SQL Server 数据类型	MySQL 数据类型	Oracle 数据类型	示例
bit	bit	bit	（none）	1
integer	int	int	number	43
decimal	decimal	decimal	number	58.63
real	float	float	number	80.62345

有时把字符类型称作 string 或 character string 类型。和数字类型不同，字符类型不再限定为数字。它们可以包括任意的字母、数字，甚至可以包括星号这样的特殊字符。当在 SQL 语句中为字符类型提供一个值时，总是需要用单引号把这个值括起来。相比之下，数字类型就从不使用引号。如下是字符类型的一些典型示例。

通用说明	Microsoft SQL Server 数据类型	MySQL 数据类型	Oracle 数据类型	示例
可变长度	varchar	varchar	varchar2	'Thomas Edison'
固定长度	char	char	char	'60601'

在第 2 个例子中，60601 可能是一个邮政编码。乍看上去，它好像是一个数字类型，因为它只由数字组成。这种情形很常见。即便邮政编码只包含数字，但通常还是把它定义成字符数据类型，因为我们不需要对邮政编码进行算术运算。

日期/时间类型是用来表示日期和时间的。就像字符类型一样，日期/时间类型也需要用单引号括起来。这些数据类型允许对所涉及的日期进行特殊的运算。例如，我们可以使用一种特殊的方法，来计算任意两个日期之间的天数。如下是日期/时间类型的一些典型示例。

通用说明	Microsoft SQL Server 数据类型	MySQL 数据类型	Oracle 数据类型	示例
日期	date	date	（none）	'2009-07-15'
日期和时间	datetime	datetime	date	'2009-07-15 08:48:30'

1.6 空值

表中每个单独列的另一个重要属性是，该列是否允许包含空值。空值表示某个特定的数据元素没有数据。按照字面意思解释就是没有包含数据。然而，空值不等同于空格或空白。从逻辑上讲，空值和空格要区分对待。在第 7 章中，我们会详细介绍检索包含空值的数据的细微差别。

许多数据库在显示带有空值的数据时，使用大写的单词 NULL 来表示。这么做是要让用户能够识别它包含的是一个空值，而不只是空格。我也会遵循这个惯例，在书中用单词 NULL 来强调这表示一个特殊类型的值。

数据库的主键不能包含 NULL 值。这是因为，根据定义，主键必须包含唯一的值。

1.7　SQL 的重要性

在我们离开关系型数据库的主题之前，为了让你对关系型数据库的优点和 SQL 的重要性有更深入的了解，我们来回顾一下历史。

回到计算机的早期时代（20 世纪 60 年代），人们通常把数据保存在磁带上，或者保存在磁盘存储器上的文件之中。使用诸如 FORTRAN 和 COBOL 这样的语言编写的计算机程序，通常读取输入的文件，并且一次只处理一条记录，最终将数据移动到输出文件中。这个过程必然是很复杂的，因为需要把该过程分解成多个单独的步骤，涉及临时表、排序以及多次数据传递，直到能够生成正确的输出。

到了 20 世纪 70 年代，随着分层和网络数据库的发明和使用，数据库取得了长足的发展。这些新的数据库，通过复杂的内部指针系统，使得读取数据更容易。例如，程序可以读取客户的记录，自动指向该客户的所有订单，然后指向每笔订单的所有详细信息。但是，基本上仍然是一次只能处理一条记录的数据。

在关系数据库之前，数据存储的主要问题不是如何存储数据，而是如何访问数据。当开发出 SQL 语言时，关系型数据库才真正取得了突破，因为它采用了一种全新的方法来访问数据。

和早期的数据检索方法不同，SQL 允许用户每次访问一大批的数据。通过一条语句，SQL 命令就能够检索或者更新多个表中的数千条记录。这就避免了很多的复杂性。当想要处理每一条记录时，计算机程序不再需要按照特定的顺序一次读取一条记录。过去需要数百行程序代码才能完成的任务，现在只需要几行代码就可以完成。

1.8　小结

本章介绍了关系型数据库的背景知识，以便你能够继续学习主要的话题，即从数据库中检索数据。我们已经讨论过关系型数据库的一些重要的特性，诸如主键、外键和数据类型。我们还介绍了数据中可能存在的 NULL 值。我们会在第 7 章中进一步讨论空值，在第 18 章中，再回到数据库维护的一般性主题，并且在第 19 章中介绍数据库设计。

为什么和数据库设计相关的所有重要的主题，都放在了本书后边去介绍？简而言之，采用这种方法，是为了让你能够直接投入到 SQL 的使用，而不必考虑数据库设计的细节。事实上，数据库设计是一门艺术，也是一门科学。因此，在对检索数据的细节和细微差别有了一些认识之后，再来学习数据库的设计原理，会更有意义。因此，我们将暂时忽略如何设计数据库这个问题，从第 2 章开始，直接进入数据检索的主题。

第 2 章
基本数据检索

关键字： SELECT、FROM

在本章中，我们将介绍 SQL 中最重要的主题：如何从数据库中检索数据。无论是在大企业还是小企业，SQL 开发人员最常遇到的需求就是报表需求。当然，把数据放入到数据库中也不是轻松的活儿。不过，一旦数据存在于数据库之中了，业务分析师的精力就转向可供他们使用的数据财富，以及希望从所有数据中获取有用的信息。SQL 语言就有了用武之地。

本书所介绍的数据检索的重点，和现实世界中 SQL 开发人员所面临的需求密切相关。要帮助企业破解数据库的数据中所隐藏的秘密，我们还需要学习更多的 SQL 知识。

2.1 一条简单的 SELECT 语句

在 SQL 中，数据的检索可以通过 SELECT 语句来完成。无需太多解释，我们来看一条最简单的 SELECT 语句的示例：

```
SELECT * FROM Customers
```

和所有的计算机语言一样，在 SQL 中，有些单词是关键字。这些单词有特殊的意义，而且必须以特定的方法来使用。在这条语句中，单词 SELECT 和 FROM 是关键字。关键字 SELECT 表示我们开始编写一条 SELECT 语句。关键字 FROM 用来表示从哪个表中检索数据，表名紧跟在 FROM 之后。在这个示例中，表名是 Customers。

按照惯例，我们会把关键字都用大写字母印刷出来，这样就能确保它们足够醒目。

总结一下，这条语句的意思是：从 Customers 表中查找所有的列。

如果 Customers 表如下所示：

CustomerID	FirstName	LastName
1	Sara	Davis
2	Rumi	Shah
3	Paul	Johnson
4	Samuel	Martinez

那么，这条 SELECT 语句将返回如下的数据：

CustomerID	FirstName	LastName
1	Sara	Davis
2	Rumi	Shah
3	Paul	Johnson
4	Samuel	Martinez

换句话讲，它返回了表中的所有内容。

在第 1 章中，我们曾经介绍过，为所有表指定一个主键是一种通用做法。在前面的示例中，CustomerID 列就是 Customers 表的主键。我们还介绍过，有时候将主键设置为：当表中增加新的行时，主键能够自动按照数字序列产生一个顺序编号。前面的示例就是这种情况。在本书中，我们所展示的大部分示例数据，都会有一个类似的列，它既是主键，又定义为自增型的主键。按照惯例，该列通常是表的第一列。

2.2　语法注释

当编写 SQL 语句时，一定要记住两点。首先，SQL 语句中的关键字不区分大小写。单词 SELECT 等同于“select”或“Select”。

其次，可以把 SQL 语句写成任意多行。例如，SQL 语句：

```
SELECT * FROM Customers
```

等同于：

```
SELECT *
FROM Customers
```

把每一个重要的关键字作为单独一行的开始，这通常是一个好主意。当遇到较为复杂的 SQL 语句时，这会让我们更容易快速掌握语句的含义。

最后，当我们在本书中介绍不同的 SQL 语句时，经常会既给出具体示例，又给出更为一般的格式。例如，前面这条语句的一般格式如下所示：

```
SELECT *
FROM table
```

斜体用来表示通用表达式。斜体的单词 *table*，表示在这里你可以用任意表名来替代该单词。所以，当你看到在本书中的任何 SQL 语句中的斜体字时，那就是我想告诉你，请在这里放上任何有效的单词或短语。

数据库的差异：MySQL 和 Oracle

许多 SQL 实现要求在每条语句的末尾放一个分号（;）。这适用于 MySQL 和 Oracle，但是不适用于 Microsoft SQL Server。为了简单起见，本书中的 SQL 语句都没有分号。如果你使用 MySQL 或 Oracle，那么需要在语句的末尾加上一个分号。因此，上述语句应该是这样：

```
SELECT *
FROM Customers;
```

2.3 注释

当编写 SQL 语句时，我们常常想要在语句之中或周围插入注释。在 SQL 中，有两种标准的编写注释的方法。第 1 种方法是使用双中划线，由位于一行之中的任意位置的两条中划线组成。这两条中划线之后的任何内容都将被忽略并当做注释。这种注释格式如下所示：

```
SELECT
-- this is the first comment
FirstName,
LastName -- this is a second comment
FROM Customers
```

第二种格式借用了 C 语言的方法，由/*和*/字符之间的文本组成注释。/*和*/之间的注释可以写成多行，如下例所示：

```
SELECT
/* this is the first comment */
FirstName,
LastName /* this is a second comment
this is still part of the second comment
this is the end of the second comment */
FROM Customers
```

数据库的差异：MySQL

MySQL 支持双中划线和 C 格式（/*和*/）这两种注释，但是略有不同。当使用双中划线时，MySQL 要求在第 2 个中划线之后，紧接一个空格或者诸如制表符这样的特殊字符。