HZ BOOKS
华章教育

全美经典
学习指导系列

SQL 编程
习题与解答

FUNDAMENTALS OF SQL PROGRAMMING

最佳的复习资料，实用的辅助教材

与国外高校计算机水平保持同步

为考研和出国深造奠定坚实基础

Ramon A. Mata-Toledo　Pauline K. Cushman 著
胡志君 高燕林 等译

全球销售超过3000万册！

Mc Graw Hill
机械工业出版社 China Machine Press
中信出版社 CITIC PUBLISHING HOUSE

全美经典
学习指导系列

SQL编程习题与解答

FUNDAMENTALS OF SQL PROGRAMMING

Ramon A. Mata-Toledo　Pauline K. Cushman 著
胡志君 高燕林 等译

机械工业出版社 China Machine Press
中信出版社 CITIC PUBLISHING HOUSE

本书主要是根据最新国际标准SQL92，为想学习SQL语言的人们编写的。书中用大量的例题讲解了SQL命令的使用及注意事项，对于从事数据库开发和进行电子商务设计的工程技术人员来说，是一本很好SQL/92参考手册；对于自学者更是一本难得的SQL自学教材，它可以帮助你在考试中取得好成绩。

本书版权登记号：图字：01-2002-0355

图书在版编目（CIP）数据

SQL编程习题与解答/（美）托勒多（Toledo, P. A.）等著；胡志等译. -北京：机械工业出版社，2002.8

（全美经典学习指导系列）

书名原文：Fundamentals of SQL Programming

ISBN 7-111-10851-5

Ⅰ. S… Ⅱ. ①托… ②胡… Ⅲ. SQL语言－程序设计－解题 Ⅳ. TP312-44

中国版本图书馆CIP数据核字（2002）第063173号

机械工业出版社（北京市西城区百万庄大街22号 邮政编码100037）

责任编辑：华章

北京忠信诚印刷厂印刷·新华书店北京发行所发行

2002年8月第1版第1次印刷

787mm×1092mm 1/16·18.25印张

印数：0 001-5 000册

定价：29.00元

作 者 序

本书是针对想学习 SQL 语言的人们编写的。SQL 语言是用于与关系数据库管理系统进行通信的标准计算机语言，虽然不同的数据库管理系统的开发商使用不同的 SQL 版本，但我们一般还是尽可能使用通用的 SQL。为了便于说明，本书使用在 Windows 95/98/NT 下运行的 Personal Oracle 8i 作为数据库管理系统，这不仅因为该系统已被广泛使用，还因为它易于获得。读者可从 Oracle 公司的网站（www.oracle.com）免费下载该系统。本书所用的全部 SQL 代码均可在装有 Oracle 的平台上工作。此外，为了便于阅读，本书的所有源程序均可从 www.cs.jmu.edu/sqldata/ 下载。

本书可与市场上销售的大部分通用的数据库书籍一同使用，同时它是我们所推出的 Schaum 系列丛书《关系数据库习题与解答》一书的姊妹篇。

感谢 McGraw-Hill 的全体员工的支持和帮助，特别感谢我们的责任编辑 Babara Gilson。希望本书对 SQL 语言及关系数据库的介绍能为读者提供帮助。享受 SQL 吧！

目　录

第 1 章　SQL 导论及关系数据库的概念

1.1　SQL 语言

SQL 语言是用于和关系数据库管理系统（RDBMS）进行通信的标准计算机语言。SQL 标准由国际标准化组织（International Standards Organization，ISO）及美国国家标准协会（American National Standard Institute，ANSI）定义。该语言的正式名称为国际标准数据库语言 SQL（1992），这个标准的最新版本通常称为 SQL/92 或 SQL2。本书中，我们将该标准称为 ANSI/ISO SQL 标准或简称为 SQL 标准。SQL 语言最初定义于 20 世纪 70 年代早期，那时叫 SEQUEL，是 Structured English QUEry Language 的缩写。SEQUEL 首先作为系统 R 的一部分实施。系统 R 是 IBM 关系数据库管理系统的一个原型。SEQUEL 名称中的单词 English 最终被去掉，这样就缩写为 SQL。Oracle 公司（最初的名称为 Relational Software Inc.）于 1979 年推出了该语言的第一个商业版本。

大多数的关系数据库开发商支持 SQL/92，但并不是百分之百地符合这个标准。目前市场上存在一些不同风格的 SQL，因为每一个 RDBMS 开发商都试图拓展这个标准，以增加产品的吸引力。在本书中，我们尽可能使用 SQL/92 标准。但我们通过 Personal Oracle 8i（Oracle 关系数据库管理系统的 PC 版本）阐述这些特性，对于那些明显拓展或偏离标准之处，我们将指出该语言的一些专有版本的差异或兼容性。

SQL 语言的一个主要特点是：它是一个说明语言，也可以称为非过程语言。对编程者来说，这意味着不用一步一步地详述计算机为获得特定结果而需执行的所有运算，而是由编程者指明数据库管理系统需要完成的任务，然后让系统去自行决定如何获得想要得到的结果。

SQL 语言的语句或命令也称为数据子语言，通常分为两大类。每一个子语言与 SQL 语言的某一方面相关，其中一类是数据定义语言（data definition language[①]，DDL），包括一些支持定义或建立数据库对象（如表、索引、序列及视图）的语句，最常用的 DDL 语句为不同形式的 **CREATE**、**ALTER** 及 **DROP** 命令。在后面的相关章节中，我们将详细讨论每一个语句。另一类子语言为数据操纵语言（data manipulation language，DML），包括允许对数据库对象进行处理或操纵的语句，最常用的 DML 语句为不同形式的 **SELECT**、**INSERT**、**DELETE** 及 **UPDATE** 语句，这些语句也将在后面讨论。需要注意的是，在一个数据库中建

---

① 数据定义语言在 ANSI 中称为模式定义语言。

立的全部对象都保存在数据字典或目录中。

SQL 语言能够以交互方式使用或将它嵌入表单中。交互式 SQL 允许用户对数据库管理系统直接发出命令[①] 并且收到运行后的结果。当使用嵌入式 SQL 时，SQL 语句包含在用通用的语言（如 C、C++ 或 COBOL 语言）编写的程序中，此时我们将通用的编程语言称为主语言（host language)。使用嵌入式 SQL 的主要原因是为了使用附加的程序设计语言的特性，而这些特性通常不被 SQL 支持。

当使用嵌入式 SQL 时，用户不能直接观察到不同的 SQL 语句的输出，结果以变量或过程参数的形式返回。

任何可交互式使用的 SQL 指令，均可用作为应用程序的一部分，这是一条总原则。但用户必须记住，当 SQL 语句交互式使用或嵌入一个程序时，可能存在一些语法的变化。本书中我们仅考虑 SQL 的交互式形式。

因为 SQL 专门与关系数据库管理系统一起使用，所以对这种数据库有一个基本了解是必要的，这样可以更好地理解这种语言的不同特性。接下来我们将讨论关系数据库管理系统。

1.2 关系数据库管理系统

允许用户定义、建立及维护一个数据库，并对数据提供可控式存取的软件系统称为数据库管理系统（database management system，DBMS)。数据库一般指数据本身，但在一个计算机化的数据库中，DBMS 还包括一些附加组件，如硬件、软件本身及用户。用户可根据需要使用数据库管理系统提供的应用及接口来存取或检索数据。用户使用恰当的用户名及口令登录到 RDBMS 后，可与 RDBMS 进行通信。登录成功后，用户可开始对话。当用户注销或与数据库断开连接后，对话结束。附录 A 演示了如何登录到 Personal Oracle Edition 8 数据库。

在数据库中描述现实的集合称为个体域（universe of discourse，UOD)，UOD 只包括这样一些现实，它们由逻辑上具有凝聚性并与用户紧密相关的一批事物构成。因此，数据库应该面向特定的用户或为特定目的不断地设计、建立及填充。

基于关系数据模型的数据库管理系统称为关系数据库管理系统，简写为 RDBMS。在这种类型的数据库中，构成 UOD 的所有信息以关系表示。虽然关系可用数学术语来定义，为了使用方便，我们将关系视为二维表。表或关系是拥有数据的数据库对象。在本书中，术语表及关系可互换。每一个关系均由一个关系名以及列或属性的集合组成，表中的数据以行或元组的集合出现。在关系中的属性总数称为关系的度。任何一次出现在关系中的行的总数称为关系的基数。本书中的术语列及属性可互换，同样我们将术语行及元组视为同义。在老的系统中，字段及记录分别与属性及行同义。

---

① 一些作者将用在交互式形式中的指令称作命令，将嵌入指令称作语句。在本书中这两个术语可互换。

依据 ANSI/ISO SQL 标准要求，在每个关系中每一个属性列必须有一个名称，且在同一表中所有的属性列名必须不同。但在两个不同的表中，两列可以有相同名称。至于关系的度或基数，SQL 标准对一个表中属性列的总数或行的总数并未限制，但制造商往往会对属性列的总数加以限制，而不对行的总数加以限制。从操作上看，一个表不含任何行是可行的，即形成的是一个空表，但表中至少必须有一个属性列。

例 1.1 中的图 1-1 显示的是名称为 CUSTOMER_ORDER 关系的二维表的基本形式。

例 1.1

写出图 1.1 中 CUSTOMER_ORDER 关系的度及基数。

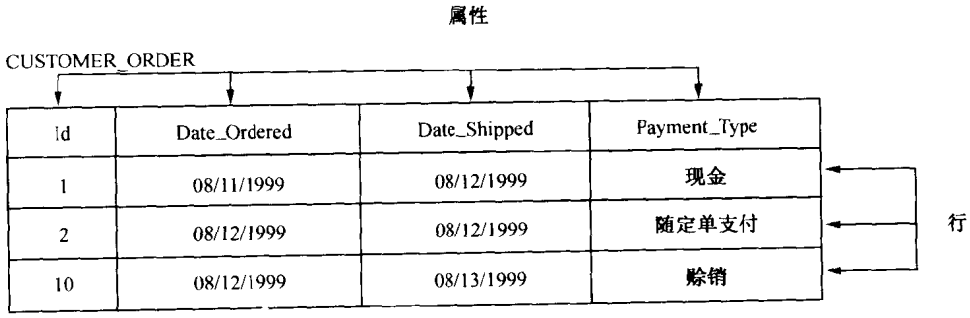

图 1-1　CUSTOMER _ ORDER 关系的二维表

图 1-1 表明 CUSTOMER_ORDER 关系由四个属性列组成：Id、Date_ Ordered、Date_ Shipped 及 Payment_ Type，表中有三个不同的行。根据前面讲的术语，称CUSTOMER _ ORDER 关系的度为 4，基数为 3。

在本书中，我们假定一个关系的每项至多有一个单一值，即每行及每列相交处最多有一个单一值[①]。对于任何给定的关系 r，r 中的任意属性 A 及 r 中任意元组 t，我们可用符号t(A)表示元组 t 在属性列 A 下的值，该值在列 A 和行 t 的交点。

例 1.2

在图 1-1 的 CUSTOMER _ ORDER 关系中，如果 t 是表中的第一行，A 是关系中的任意列，写出各个 t(A)值分别是多少？如果 t 是表中的第一行，则 t(Id) = 1，t(Payment _ Type) = 现金，t(Date _ Ordered) = 08/11/1999，t(Date _ Shipped) = 08/12/1999。

在 CUSTOMER _ ORDER 关系中，每一个属性列有一个相关的域，通常表示为 Domain(属性)。域等同于出现在关系 r 中的特定列的值的集合，即对于关系 r 中的任一元组 t 及任一属性 A，t(A)必须是 Domain(A)的一个元素，用数学术语中的集合理论，可表示为 t(A)∈Domain(A)。

① 这保证了关系属于第一规范式，即 1NF。

例 1.3

确定图 1-1 的 CUSTOMER _ ORDER 关系表中不同属性的可能域。

在本例中，我们假定属性 Id 的域为正整数集，属性 Date _ Ordered 及属性 Date _ Shipped 的域为有效日期集，属性 Payment _ Type 的域则为包括以下值的有限长度字符串的集：{现金、赊销、邮政汇票、支票、随定单支付、信用卡、借记卡、未知}，由此我们可以如下表示这些域：

Domain (id) = 正整数集

Domain (Payment _ Type) = {现金，赊销，邮政汇票，支票，随定单支付，信用卡，借记卡，未知}

Domain (Date _ Ordered) = Domain (Date _ Shipped) = 有效日期集

注意，在 CUSTOMER _ ORDER 关系中，对每行 t 及属性 A 而言，t (A)是它对应域的一个元素。

1.3 关系的候选码及主码

在关系模型中，码是一个基本概念，在数据库中它可提供检索元组的基本机制。假设有一个关系 r 及其属性 A_1，A_2，A_3，……，A_n，只要 K 满足以下条件，我们可将这些属性的任意子集 K = {A_1，A_2，……，A_k} 称为候选码：

1. 对于关系 r 中任意两个不同的元组 t_1、t_2，存在子集 K 的属性 B，且 $t_1(B) \neq t_2(B)$，这表明 r 的两个不同元组在 K 的所有属性中均无相同的值，这个条件就是码的惟一性特性。
2. K 的子集中不存在满足惟一性特性的子集 K′。换句话说，删除 K 中的任何元素都会破坏惟一性特性，这个条件就是码的最小性特性，该特性保证了组成码的属性个数最少。

在一个关系 r 中，存在多个候选码，故可将其中的任意一个候选码定为关系中的主码(primary key，PK)。主码的值可用作该关系的寻址机制。主码一经选定，其他候选码有时也称为替代码。对每个表而言 RDBMS 只允许有一个主码。主码可由一个单一属性组成(单一主码)，也可由多个属性组成(组合主码)。在本书中，我们将对主码中的属性用下划线标志出来。在例 1.1中，属性 Id 是 CUSTOMER _ ORDER 关系中的主码(注意 Id 已经有下划线)。既然 Id 已是 CUSTOMER _ ORDER 关系中的主码，那么没有两份订单具有相同的 Id 值。

因为主码用于确定一个关系中元组或行的惟一性，故构成主码的属性值不可为空(NULL)。这样对主码就产生了附加的限制，称为完整性约束。一个空值表示无信息、未知数或不恰当的数据。读者必须记住一个空值既不是一个零值，也不代表计算机中的一个特定值[①]。

① 在 SQL 中，空值和其他值之间的大多数比较是通过定义未知而非真或假来进行的。

例 1.4

DEPT 表及一些行如下所示，试解释这些行能否插入 DEPT 表。

DEPARTMENT	NAME	LOCATION	BUDGET
10	Research	New York	1500000
	Accounting	Atlanta	1200000
15	Computing	Miami	1500000

DEPT

DEPARTMENT	NAME	LOCATION	BUDGET
20	Sales	Miami	1700000
10	Marketing	New York	2000000

10	Research	New York	1500000	← 该行不能插入 DEPT 表。它违反了码的惟一性特性，因为表中已存在部门 10
	Accounting	Atlanta	1200000	← 该行不能插入 DEPT 表。它违反了码的完整性约束，因为部门码不能为空 (NULL)。
15	Computing	Miami	1500000	← 该行可插入 DEPT 表。因为它不违反任何约束。

1.4　外码

因为具有相同基础域的列也可用在数据库的关系表中，外码(foreign key，FK)的概念使 DBMS 维持了两个关系的行间或同一关系的行间一致性。该概念可正式定义如下：假设在同一数据库中已知有关系 r_1 及关系 r_2[①]，如果同时满足以下两个条件，则关系 r_1 的 FK 属性组可称为 r_1 的外码：

- FK 中的属性与关系 r_2 中已定义为 r_2 的 PK 属性组具有相同的基础域，则 FK 是以关系 r_2 的 PK 属性为参照。
- 关系 r_1 中任何元组的 FK 值或为 NULL，或为关系 r_2 中一个元组的 PK 值，两者必居其一。

在实际操作中，外码概念保证了：关系 r_1 的元组若引用关系 r_2 的元组，关系 r_2 的元组必须已经存在。对外码的这个约束称为参照完整性约束。一些作者将包含外码的表叫子表，包含参照属性的表叫父表。应用这些术语，可以称一个子表的每一行的 FK 值或为 NULL，或与父表的一个元组的 PK 值(或 UNIQUE 值)相对应。

① 按照外码的这个定义，关系 r_1 和 r_2 可以是同一个关系。

例 1.5

假设下表中属性 EMP _DEPT 是 EMPLOYEE 表的一个外码，其中 EMPLOYEE 表以 DEPARTMENT 表的属性 Id 为参照。请指出下面给出的各行是否可插入 EMPLOYEE 表。

ID	NAME	LOCATION
10	Accounting	New York
40	Sales	Miami

DEPARTMENT

EMP ID	EMP_NAME	EMP_MGR	TITLE	EMP_DEPT
1234	Green		President	40
4567	Gilmore	1234	Senior VP	40
1045	Rose	4567	Director	10
9876	Smith	1045	Accountant	10

EMPLOYEE

9213	Jones	1045	Clerk	30
8997	Grace	1234	Secretary	40
5932	Allen	4567	Clerk	NULL

该行不能插入 EMPLOYEE 表。它违反了码的参照完整性约束。在部门表中没有部门 30。

它没有违反任何约束，该行可以插入 EMPLOYEE 表。

该行可插入 EMPLOYEE 表。在 EMP_DEPT 列中NULL 值是可接受的。NULL值可能表示该雇员尚未分配到某一部门。

注意：在上例中，我们使用关键字[①] NULL 表明部门的缺乏状态，而在例 1.4 中，DEPARTMENT 的输入保留了空白。读者必须注意到在某些系统中，允许以两种方式表示空值，而在另一些系统中，只允许以关键字 NULL 表示空值。

1.5 关系运算符

关系运算符是一组能够使我们对数据库的表进行操纵的运算符，它们具有封闭特性(closure property)，因为能根据关系操作产生新的关系。当某个关系运算符用于操纵一个关系时，可称该运算符被应用于关系。在本部分仅讨论选择、投影及等值连接关系运算符。在第 2 章，我们将说明在 SQL 中如何使用这些运算符以及一些辅助的关系运算符。

① 关键字指对系统有特殊意义的单词，不能在特定内容之外使用。

1.5.1　选择运算符[①]

当选择运算符用于关系 r 时，这个运算符会产生另一个关系，它的行是满足指定条件的在关系 r 中行的子集，所产生的关系和 r 具有相同的属性。我们可以给这个运算符一个比较正规的定义：假设 r 是一个关系，A 是 r 的一个属性，a 是 Domain (A)的一个元素，在属性 A 上 r 的选择，即产生 r 的 t 元组子集且 t (A) = a。在 A 上 r 的选择可表示为 $\sigma_{A=a}(r)$。下例说明了选择运算符是如何工作的。需要注意的是：选择运算符是一个单目运算符，即一次只对一个关系进行运算。

例 1.6

在上例的 EMPLOYEE 关系中，找出所有在部门 10 工作的雇员的信息。

因为我们需要检索所有在部门 10 工作的雇员信息，所以必须确定 $\sigma_{EMP\_DEPT=10}$(EMPLOYEE)。注意此例中关系 EMPLOYEE 的元组需满足的条件是 t (EMP _DEPT) = 10。产生的表如下所示：

$$\sigma_{EMP\_DEPT=10}(EMPLOYEE)$$

EMP ID	EMP_NAME	EMP_MGR	TITLE	EMP_DEPT
1045	Rose	4567	Director	10
9876	Smith	1045	Accountant	10

1.5.2　投影运算符

投影运算符也是一个单目运算符，不同之处在于：选择运算符选择关系中行的子集，而投影运算符选择列的子集。该运算符的正式定义如下：关系 r 对其属性组 X 的投影可表示为 π_X (r)。它是一个新的关系，可以通过首先排除关系 r 中未在 X 中指出的列，然后排除任意重复元组来获得。在下面的例子中，我们将对此进行说明。

例 1.7

在下面的 DEPARTMENT 表中，指出不同部门分别位于何地？根据这个回答，我们能否说出在这个表中有哪些不同的地点？

① 选择运算符也可称为 Restrict 或 Select 运算符，我们不使用单词 select 是为了避免与 SQL 命令中的 select 相混淆。

DEPARTMENT

ID	NAME	LOCATION
10	Accounting	New York
30	Computing	New York
50	Marketing	Los Angeles
60	Manufacturing	Miami
90	Sales	Miami

由于我们需要决定目前在 LOCATION 列中出现的不同值的数量，因此需在 DEPARTMENT 表的 LOCATION 属性上做投影，即需要找出 $\pi_{LOCATION}$(DEPARTMENT)，此时，r= DEPARTMENT, X= {LOCATION}，可得到如下关系：

$\pi_{LOCATION}$(DEPARTMENT)

LOCATION
New York
Los Angeles
Miami

注意，在结果表中，New York 及 Miami 只出现了一次。也可观察到 LOCATION 是此表中的惟一属性。

总体说来，关系在属性 LOCATION 上的投影，不能告诉我们 DEPARTMENT 表中目前出现的地点总数，因为重复值已被去掉。在投影关系中只有三个地点，而 DEPARTMENT 表中地点的总数是 5。

例 1.8

使用上例的关系表，指出不同的部门及其所在地。

此例中，X= {NAME, LOCATION}，r=DEPARTMENT，因此可得出以下关系：

$\pi_{NAME,LOCATION}$(DEPARTMENT)

NAME	LOCATION
Accounting	New York
Computing	New York
Marketing	Los Angeles
Manufacturing	Miami
Sales	Miami

由于部门名称及其所在地的组合是惟一的，因此结果关系中不含重复值。注意，(Manufacturing, Miami)及(Sales, Miami)是不同的元组，同样，位于纽约的不同部门也是不同的元组。

1.5.3　等值连接运算符

等值连接运算符[①] 是二元运算符，它可将两个关系(不一定相同)组合到一起。通常，这个运算符通过两个关系中的共同属性将它们连接到一起。即来自第一个关系的元组与第二个关系的元组对某一共同属性 X 有等值时，将它们并置可得到全部元组的连接。从数学上可如下定义：设关系 r 含有属性组 R，关系 s 含有属性组 S，并且 R 及 S 有一些共同的属性[②]。设关系 R 和 S 共同的属性组为 X，可表示为 $R \cap S = X$。r 与 s 的连接(可表达为 r Join s)是一种新的关系，其属性是 $R \cup S$ 的元素。此外，对 r Join s 关系中的每一元组 t，需同时满足以下三个条件：(1)关系 r 中的某一元组 t_r，满足 $t(R) = t_r$；(2)关系 s 中的某一元组 t_s，满足 $t(S) = t_s$；(3) $t_s(X) = t_r(X)$。下例对该运算符如何起作用进行了说明。

例 1.9

根据下面所示的两个表的共同属性 DEPT 连接这两个表。写出该操作结果可满足哪些可能的用户请求？这两个表能用属性 ID 连接吗？

DEPARTMENT

ID	DEPT	LOCATION
100	Accounting	Miami
200	Marketing	New York
300	Sales	Miami

DEPARTMENT

ID	NAME	DEPT	TITLE
100	Smith	Sales	Clerk
200	Jone	Marketing	Clerk
300	Martin	Accounting	Clerk
400	Bell	Accounting	Sr. Accountant

在本例中 DEPT 是两表的共同属性，这两表的连接可表示为 DEPARTMENT **Join** EMPLOYEE(如下表)。由于两个表均有属性 ID，为了避免 DEPARTMENT 表中的属性 ID 与 EMPLOYEE 表中的属性 ID 相混淆，在连接两表时，需要将此属性与它相应的表名一起列出。注意：这一点与前面所讲的“表的列名称不能相同”的相一致。还有一点需注意，两个表共同的列连接后，不再重复给出。该连接操作的结果可以满足用户“显示有关雇员的所有信息及他们所属部门 ID、部门名称及部门所在地”的请求。

① 此类型的连接也称为自然连接。

② 共同属性并不要求两表中有相同名称的属性，但它们的基本含义和域必须相同。

DEPARTMENT Join EMPLOYEE

DEPARTMENT ID	DEPT NAME	LOCATION	EMPLOYEE ID	EMPLOYEE NAME	TITLE
100	Accounting	Miami	300	Martin	Clerk
100	Accountong	Miami	400	Bell	Sr. Accountant
200	Marketing	New York	200	jones	Clerk
300	Sales	Miami	100	Smith	Clerk

DEPARTMENT 表及 EMPLOYEE 表不能由属性 ID 连接，因为 DEPARTMENT 表的属性 ID 与 EMPLOYEE 表的属性 ID 具有不同含义，一个是部门的 ID，一个是雇员的 ID。这表明两个表不能仅根据它们的属性具有相同的名称而将两个表连接起来。两个表必须根据它们的共同属性连接，这些共同属性必须有相同的域及相同的意义。

在第 2 章中讨论关系运算符在 SQL 中的执行情况时，将对此进一步讨论。

1.6 属性域及其执行

如前所述，属性域定义了一个表中包含的列值的特性。在任一 RDBMS 中，任一给定的属性域用一个数据类型来实现。标准 SQL 语言命名并定义了一系列的基本数据类型。虽然大多数的 RDBMS 开发商均支持这些数据类型，但它们在各个开发商的实现细节上并不相同。因此建议用户查询相关的 SQL 参考手册，以确定 RDBMS 开发商实现的数据类型特点。表 1-1 列出了一些基本的 SQL 数据类型及几个 RDBMS 开发商的实现。

表 1-1 某些标准 SQL 数据类型及某些 RDBMS 开发商的实现

标准 SQL	Oracle	Access	DB2
Character (n)：n 是指字符数	Char (n)：固定长度的字符，最大值为 255 个字符	Text：固定长度的字符，最大值为 255 个字符	Character (n)：与 Oracle 的 Char (n)相同
Character varying (n)：n 为可存储于列中的最大字符数	Varchar2 (n)：可变长度的字符。最大值为 2000 个字符	Text：可变长度的字符，最大值为 255 或 Memo 可变长度的字符，最大值为64 000	Varchar (n)：与 Oracle 的 Varchar2 (n)相同
Float (p)，p 是数字的总个数	Number：数的大小在 1.0×10^{-130}和 38 个 9 后接 88 个 0	Single 或 Double：取决于数据值的范围	Float：与 Oracle 的 number 相同
Decimal (p,s)：至少有 p 位数字，带有 s 位小数位	Number (p,s)：p 的范围在 1 至 38，而 s 的范围在 - 84 至 127 之间	Integer 或 Long Integer：取决于数据值的范围	Integer：与 Oracle 的 number (38)相同

数据类型 character (n)或 character varying (n)的列(此处的 n 指可存储在列中的最大字符数)通常用于含文本的数据或不参与计算的数据。这种数据类型的例子有：名称、地址、社会保障号码及电话号码等。character (n)与 character varying (n)数据类型之间的主要不同之处在于它们如何存储短于列最大长度的字符串(字符的顺序)。当一个字符串少于 n 个字符