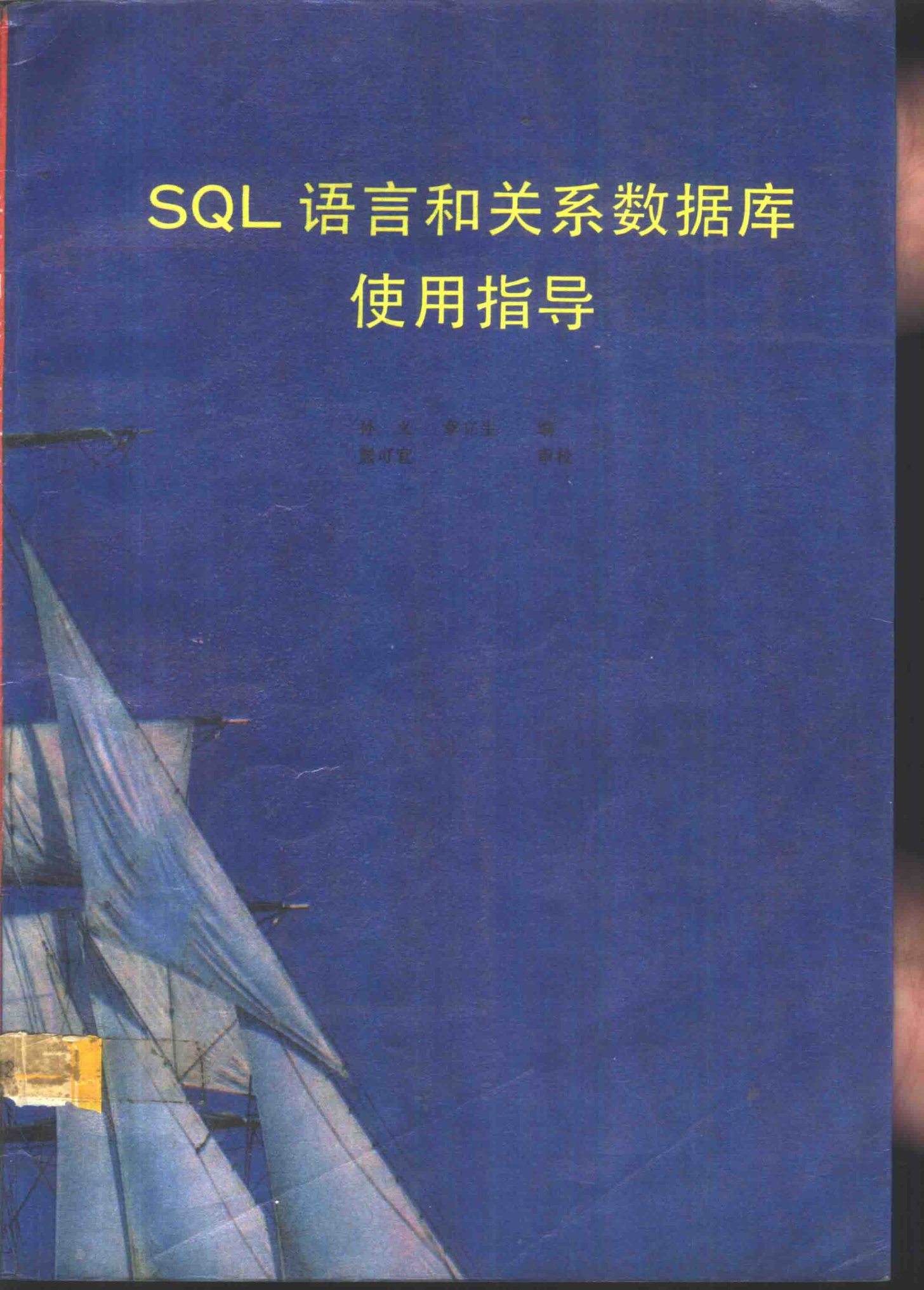
SQL语言和关系数据库使用指导

孙义 李行生 编
顾可宏 审校



SQL 语言和关系数据库使用指导

孙 义 章立生 编
熊可宜 审校

目 录

第一章　关系数据库

1.1 历史背景

商业用的数据库管理系统60年代后期问世．IBM的层次数据库管理系统IMS（信息管理系统）是其中之一．

在数据处理早期，只有很少的独立标准函数和过程．六十年代有了较快的发展．随着应用和越来越多，所涉及处理的信息数量也越来越大．因此需要将数据集中存贮，并能快速、方便地访问它们．

这样的集中存贮有下列目的:

- 减少维护数据的人数．
- 允许数据共享．
- 允许许多人共享程序．
- 引进数据结构和信息恢复的标准．
- 避免不一致性和错误，例如：双重目录顺序或双重贷特存贮．

由集中式存贮引出的新问题包括:

- 数据更大的复杂性和独立性．
- 处理数据和为用户服务的程序将更大更复杂．
- 在多用户应用和条件下对安全性和个人的保密性提出新的要求．
- 计算机和软件依赖性越大，产生的操作问题和系统错误就越严重．但这样的优点是人们总是可以抱怨是计算机将一切事情搞糟了．

物理的解决办法是大型机．分散的用户使用远程终端，但缺乏局部处理能力．主CPU(中央处理单元)完成全部运算．

从管理的角度讲，集中式的数据处理意味着将管理分成几部分来组织．这样的部分有三组成员:

- 数据库管理员（以下简称DBA），负责全部管理，诸如：创立数据库，安全性管理以及决定开发策略．
- 应用程序员：负责建立、维护和开发程序用来满足结构化数据处理的需要．
- 系统操作员：负责保证数据库管理能在适当的软件和硬件条件下运行．

数据处理（或信息系统）部分以外，终级用户完成由数据库管理员和程序员控制的应用．

70年代和80年代出现的微型机和个人计算机(PCs)使得集中式出现相反的发展倾向，即新的理论“一人一台机器”．

1981年IBM PC的出现使得这种低造价的小机器成为办公室、学校、医院、工厂和

其它地点的标准固定设备。PC 机很快地成为功能强大、正式的工具，从某种程度上来说应感谢诸如：电子信件、字处理、图形和写程序、程序语言和数据库管理系统这样的广泛应用。

由此带来的问题是如何使 PC 机之间相互通信并同主机的资源共同工作。不久局部网(LAN)成为广泛使用的方法。

基本问题之一是使不同类型的软件和硬件之间实现通讯。一种解决的方法是实现由 ISO 和 IEEE 共同提出的 OSI(Open System Interconnection)标准化模型。IBM 已经开发了自己的一套规则（系统应用结构或 SAA 概念）。它允许应用程序运行在大型计算机到微型机、工作站和个人计算机等各种设备上。开发者不必顾虑用户在特定条件或环境下所使用的设备。

SAA 概念允许开发者在 PC 机上写应用程序而在大型机上运行或者相反。它允许从一种计算机传送到另一种计算机并可将数据存在不同的机器（甚至不同类型的机器）上。换句话说，SAA 概念就是通过一个组织来使软件和硬件全部一体化，从而简化了开发、使结果、计划、训练协调和结合在一起。

1.2 其它类型的数据库

IMS（信息管理系统）是层次数据加管理系统。它有两个主要特点。一个是用于写数据库软件的语言和数据物理组织之间的密切关系。另一个是在各个数据集（表）之间的层次关系。

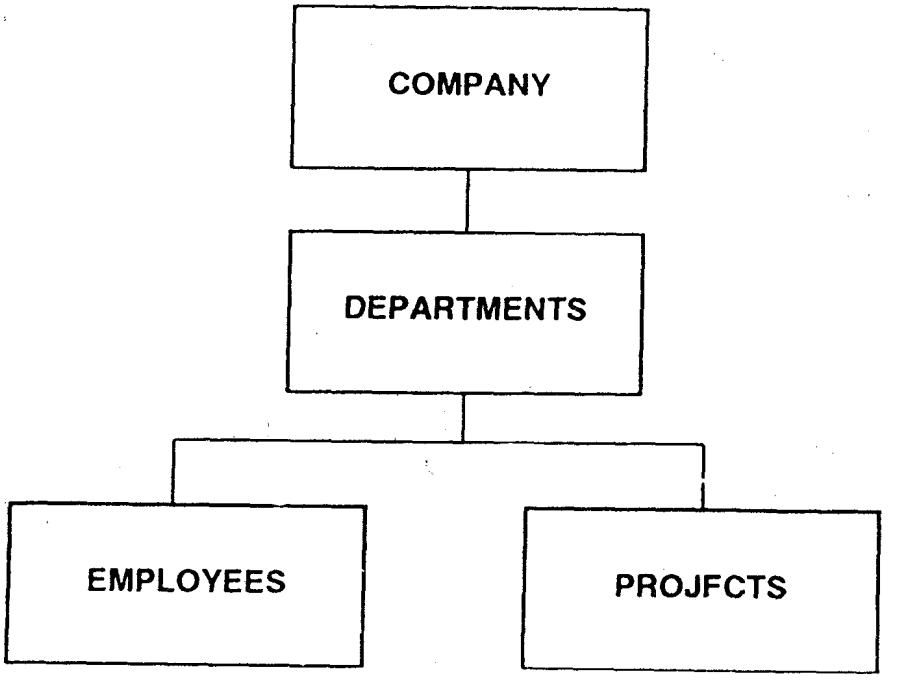


图 1-1　公司的层次结构

层次数据结构包含在各个数据集之间形式化了的上下层关系，如图 1-1 反映公司的组成关系．关系依赖于数据集通过它们地址（指针）的联系．其结果就象有着固定通信格式和已排好顺序的军事命令链．

层次结构存在着严重问题，它防碍对存放数据操作的灵活．这种数据和指针结构决定了所能完成的操作．如果数据库没有准备进一步建立关系，那么没有建立必要指针的关系就不能使用．

在层次结构中，命令链是固定的．例如，找到在一个部门的所有项目，就必须遵循已定义好的关系指针．

层次数据库的特点僵化（因此也限制查询选择项）在网络数据中在某种程序度上得到缓解．我们也可以不沿着一个命令链（从上到下）而交叉查询．但我们仍然要使用唯一建立的路径．

两种数据库的共同特点是用户不能增加或删除数据．他或她也必须改变链接数据的指针．因此，使用层次和网状数据库不仅意味着与数据打交道，同时也与结构和物理元素打交道．

这样的数据库的另一特点是既不灵活，用户界面也不友好．用户必须自己写程序来完成没有预先定义的查询．

然而，正因为不必决定查询路径，僵化的结构却能带来快速的执行．高速度就是 IMS 能在现今广泛应用的原因之一．

此外，逻辑数据库（例如，数据表）与物理级紧密相联，也即与下层的文件和控制系统相联．这个关系链意味着在数据存贮和优化或加速查询方面的高级控制．

物理与逻辑层的密切关系也影响了可移植性．层次数据库只能运行在特定结合的硬件和操作系统上．

也有几种方法将物理和逻辑层分开．例如，CODASYL 组设计了一个数据库体系，它将被定义的表和结构的语言与数据的物理结构之间的关系描述成接口．这个工作 1971 年出现．后来由 ANSI／SPARC 研究组推出网络数据库标准．它满足某些逻辑数据独立性的要求，也即在物理和逻辑数据之间存在接口的结构．它也提出在数据库控制系统和计算机之间也有标准接口．

分成两级的思想就是允许用户完成查询和其他任务而无须考虑机器的存贮方法或介质，用在物理级的指针类型．因此这种结构能容易地更换硬件和软件以满足快速变化的需要．这里强调的是灵活性而不是最大的速度．

1.3 特性

60 年代后期和 70 年代早期，IBM 研究中心的工作人员在 San Jose 机上开发了叫作 System R 的原型关系数据库管理系统．它就是现行的诸如 SQL／DS 和 DB2 等 IMB 产品的前身．

E.F.Codd 在他的题为“用于大型共享的数据体的关系数据模型”的文章中(CACM,Vol.13No.6,June1970)，阐述了关系数据模式的基本思想．这个理论大体上是数学上的．它描述了 ANSI／SPARC 体系扩展的数据库结构和能在这结构上操作的语言．

关系数据库的思想是一个动态的、灵活的和与用户有良好接口的结构．用户应该能不用写程序就可以管理信息．象 ANSI／SPARC 结构这样系统的主要特点是物理和逻辑级严格分开．这样作是必要的以保证用户和数据库控制系统能集中于数据．它将指针和数据库的物理组织的管理都由另外的部分处理．第十一章将更详细地讨论这点．

象先前提出的这样的关系数据库由表构成．但现在有两点新的要求:

·数据库结构必须由表而且只能由表组成．结构必须是平面的，也即只有表的内容才能决定何种查询和改变是可能的．换句话说，结构不提供物理安排、指针结构或其他硬件关系的无素．

·定义和处理必须是独立的物理存贮结构．关系数据库模型是逻辑结构．它的描述与其物理工具不应有联系．这个思想就是允许修改、移植和分布处理．

数据库本身应该有它自己的控制系统，关系数据库管理系统或 RDBMS，其操作只基于逻辑结构．作为指针不再决定何种检查询是可能的，只有数据内容决定数据控制系统的扩充．解决的方法是形式化的"数据字典"或"系统目录"．它是一个包括原始数据库结构概况的数据库，例如，它的表、域、数据类型说明．数据字典就象一个长报告的总结或书的目录表．

关系的 DBMS 基于数据库的逻辑说明来决定如何找到数据．系统建立和维护描述在数据（在以表形式存在）的 DBMS 视图、数据字典、物理存贮的数据和计算机操作系统之间联系的"映射"或"指针"．RDBMS 能够维护在理论上应该是独立的逻辑和物理两级数据库．

DBMS 评估一个命令并决定一个可能的查询路径，它调映射．由映射调用 OS 的 I／O 进程并允许它们去取物理数据．

物理和逻辑两方面分开意味着 DBMS 处理全部与安全性有关的事物，以及完整性和一致性（例如，保护两个人在相同时间修改同样的数据）．此外，DBMS 本身解释一个查询，通过表找到一个选择路径并与映射过程通信．

DBMS 必须分析每一个查询的事实限制了它在实际中的应用．"在跳之前先看"是句好格言，但减慢了操作速度．许多早期关系数据库太慢．额外的分析和翻译占用时间并增加了系统的复杂性．只有最近的 DBMS 速度快才真正发挥了其灵活性的优势．

1.4 SQL

Codd 除了定义了数据库的逻辑结构还定义了关系查询语言．到 1976 年，这种语言变成 IBM 的 SQL(结构查询语言)．

1986 ANSI(美国国家标准所)出版了 SQL 标准 X3.153．它使得 SQL 向着几个非关系型的数据库接口发展．ANSI 继续改进和扩充 SQL 标准并允许各个 SQL 之间互换的能力(通过 SQL 访问组)．更详细的信息请看 C.J.Data,A Guide to the SQLStandard,Znd ed (Reading,MA:Addison-Wesley,1989)．

1.5 工具

1979 年 Oracle 公司发行了第一个商用关系数据库 Oracle 版本 1．

IBM 继续开发 System R，最终出现两个商业化软件产品．他们是 SQL／DS（用于

9370 机器和 VM／CMS 操作系统)，于 1982 年出版；1985 年出版的是 DB2(用于 3090 大型机和 MVS 操作系统).

与 IBM 开发的工具(DB2 和 SQL／DS)只能用于它本身的大型机，相反 Oracle 开发的产品却能用于各种机器和操作系统．公司首先设计的软件用于 IBM 大型机和 DEC 计算机，但后来的产品可用于几乎各种计算机和操作系统，包括 MVS 和 VM．Oracle 也为 UNIX，MS-DOS 和 OS／2 开发了版本．在 IBM 和 Oracle 的产品中一般都使用 SQL 作为查询语言的标准．因此 SQL 即是 ISO 和 ANSI 的标准，也成为关系数据库语言的商业标准．

本书使用 SQL／DS 和 Oracle 作为例子．两个产品相似的开发产生相似的功能（对表创建和数据处理基本相同)．我们主要集中于 SQL 内核．但 SQL／DS 和 Oracle 也有差别，代表着 SQL 的两个方面．我们将阐述标准 SQL，但也讨论工具间的差异．

1.6 定义

数据库是为一定目的而组织好的一组数据．例如，它能是目录控制、个人的（人类资源）管理，顾客服务、工资、项目管理或公司的生产计划．它也能传送火车或飞机预定、医院管理、或医学或立法记录保存．任何大小类型的单位都能从数据库中有组织的数据中获益．

关系数据库是“将数据作为一组表格呈现给用户的数据库”．(Date,Introduction to Database Systems,Vol 1,5th ed,Addison Wesley,Reading,MA,1990)．晕个定义并没有说明表格之间有特殊的关系，但他们是数据库工作的一个重要方面．我们强调“呈现给用户”因为它反映了逻辑结构和物理存贮之间的区别．不考虑数据是如何存的，它总是呈现给用户表的形式．关系数据库是逻辑结构．也即，用户通过 SQL 与关系数据库打交道时，数据以物理地存贮于一个表结构的形式出现．当然实际上它是一位一位地存在介质上而并不是存在表上．

图 1-2 表示了一个（逻辑的）表的例子．它由“NAME”“ADE”和“SEX”三列组成．三个头是变量名．如图中出现的那样，每个变量可以包含许多相同类型的数据．表的结构定义构成表的变量的名字和类型．

一列通常称为一个字段．在图 1-2 中，值 33，23，44，12 和 33 是字段“AGE”的值，而 John Smith 是字段“NAME”的值．

NAME	AGE	SEX
John Smith	33	M
Ursula Peters	23	F
Morton Berger	44	M
Peter Olson	12	M
Nina Hagen	33	F

图 1-2 样本表

由字段组成的行形式的单元叫作记录．表中的记录数随时间而变化，给定的数目叫表的基数．

一个记录定义为给定的元素，如人的特征由 NAME＝"John Smih"，AGE＝"33"，和 SEX＝"M"组成．究竟什么是计算机要保存的现代人的数据呢？记录集合定义了表的名字．在这里表是"顾员表"．

如同所有数据库一样，关系数据库表结构也要遵守一定的规则．下面几点是特别重要的．

·字段名必须是唯一的．换言之，一个表不能有两同名的字段．

这个要求的原因是明显的．如果有两个"SEX"字段，那么象"找出表中所有男性名子"这样的查询就无法进行．

·字段的顺序不起作用．我们不能说："显示第二列"．这个要求与关系数据库的结构是逻辑的这样事实密切相关．当然不能保证所有的数据库管理员都将"AGE"放在第二个域．此外，不能禁止 DBA 在表中删除或交换列，或者插人一个新的列．因此象这样的查询"从列 2 中选中所有列 3＝'M'的项"是无意义的．我们必须用名子来引用字段．

·至少一个字段或由多个字段组合可以唯一地指定一个记录．也即不能有两个相同的记录．在图 1-2 中"name"提供唯一的标识．

实际上，一个人的姓并不能唯一标识记录，因为表中可以包含两个同姓的人（诸如 Bohuslaw Miropowicz 和 Stanislaw Miropowicz 或 John Smith 和 Fred Smith）．规则规定至少一个组合字段的值能唯一地标识每个记录．例如，一人可以将"name"和"age"两字段结合而得到唯一的 ID．两个同姓同名又同年龄的人的机会较少，但还是存在的．为了避免由于这种偶然而引起麻烦，我们常常引人一个人造的列以保证每一个记录的唯一性．例如，它可以是包含一个连续的数字系统（顾员号码），社会安全号、帐号或其他标识代码．这些额外的列本身并没有什么意义，它仅用于标识的目的．

同一表中可能存在多个唯一性的标识码．例如，他们可以是一个内部号码，一个社会安全号码，或一个电话号码和名字的组合，或是记录中所有字段的组合．注意，必须至少存在一个这样的标识或表中包含标识记录．

唯一标识（一个字段或多个字段的组合）称为主关键字．如果一个表有几个唯一的标识，它们都是主关键字的候选关键字．我们因此称他们为"候选关键字"．我们可以选择其中的一个来作为主关键字．

我们因此可以说规则要求表中必须有一个主关键字．

·各个记录的顺序不起作用．符合这一要求的参数与字段的顺序相似．这再一次保证了在关系表中物理和逻辑上数据的独立性．它必须可以插人删除记录而不影响以后的操作和应用．

·所有的字段必须是"原子"的，也即他们不能有相同类型的多个值．例如，如果为某人在相同字段中记录两个银行帐号结算（如支票和存款），那么这个字段的取值就是非原子的．

总结这五点规则，数据库是一个由唯一定义了的表的集合，这些表由随时间改变数目、而取得唯一的标识的原子字段组成．这个定义虽然正确，但听起来使人难于接受，或象个无法解决的社会或环境问题．

1.7 主要的样本数据

在这本书中我们将提到一个主要的样本数据库．因此我们在这里简单介绍一下．它描述了一个在纽约州的公司．这个公司出售运动衣给俱乐部、学校、零售商、训练馆、以及其它消费者．全州都有该公司的推销员，他们很少光顾总部，与公司的其他部门没有任何关系．

主任办公室完成全面管理并保存全部产品库存清单．

顾员收到每月的薪水（基本工资），和销售奖金．

主任办公室处理全部计算机保存的顾客、顾员、定单和库存的数据库．各个部门有一个终端与主系统相联．

网络允许各部门从中心数据库取得各部门所需使用和处理的信息．

因此这个系统是高度集中化的．更一步的分布是在考虑处理进一步增加和新的应用后再处理．

主任办公室在它的数据库中保持下列信息；（如图 1-3）：

Employees	Customers	Orders
Employee_no	Customer_no	Order_no
First_name	Company_name	Customer_no
Last_name	Street	Employee_no
Street	Zip_code	Received
Zip_code	Telephone	Shipped
Telephone	Type	
Date_hired		
Type		
Department		
Base_pay		

Order_specs	Inventory	Zip_codes
Order_no	Item_no	Zip_code
Item_no	Item_name	City
Quantity	Quantity	
	Price	
	Order_level	

图 1-3　样本数据库表结构

顾员数据：　名子、地址、电话号码、雇用日期、部门、基本工资和类型(Dealer,

Manage，或 Salesperson)

顾员数据: 名子、地址、电话号码和类型(Club,Exersise studio,Retalier 或 School)

定单数据: 收到日期、发货日期、定购的项目、项目序号和推销员.

库存数据: 项目名、价格、货存号和级别序号，

到此，我们不再讨论选择表和字段的原因．附录 A 详细地说明了数据库内容．

第二章　数据定义

一个表是由一个表名和一组数据元素描述的．每个元素由一个名字和类型来说明．这章首先描述数据元素的各种类型．然后介绍 SQL 中定义和改变表结构的工具．最后一节简要地讨论 SQL 的注释和同名的使用．

2.1 数据类型

定义一个表结构，必须给每个元组指定数据类型．表 2-1 列出了最普通的数据类型和它的取值范围．

表 2-1　一般的 SQL 数据类型

CHAR	指定长度．例如，"name CHAR(25)"说明变量name是由不超过25个字符组成．缺省长度（如果不指定）为 1．最大长度通常为 250 个．在大多数工具中写作 CHARACTER．
SMALLINT	一个整数变量．取值范围为-32,768到+32,767(两个字节)
INTEGER	整数变量．取值范围为-2,147,483,648到+2,147,483,647（四字节）
DECIMAL	十进制数．符合文法 DECIMAL[(P[,S])]或 DEC[(P,S)]，这里 P(Precision) 是数字的有效位数，取值为 1 到 15．S(Scale)是数字的位数可以取从 0 到 15-P 的任何值．典型的例子是 DEC(15,0),EDC(7,2)，或 DEC(0,15)．即没有 P 也没有 S 的缺值为 DECIMAL(5,0)
FLOAT	绝对值从10^{-308}到10^{+308}（八字节）的实数．

我们采用 SQL／DS 中的数字数据类型名字．Oracle 可以识别指定的类型和名字（例如，作对话时），但它只使用 NUMBER 或 NUMBER[(P[,S])]来完成操作．NUMBER 包含上述的数据类型．使用 FLOAT 时，NUMBER 取值范围从-1E128 到+1E128，而 INT 和 DEC 的有效位 P 的范围是从 1<P<38．ANSI 标准也支持 REAL 和 DOUBLE PRECISION 两种数字变量类型．

ANSI 标准的唯一的字串类型是 CHAR．大多数工具有 VARCHAR 类型，定义如下:

VARCHAR　　CHAR有可变的长度．它象CHAR那样指定最大长度，如:
remark VARCHAR(1000)

具有 VARCHAR 类型的变量可包含 32,767 个字符（在 SQL／DS 中）或 65,535 个字符（在 Oracle 中称为 LONG）．在 VARCHAR 存贮项，DBMS 只保留串的当前长

度．如果长度增加，则增加部分存在其它地方并通过指针连起来．

VARCHAR 变量有一些限制．不能用它们作为索引或用它们作为簇列．但可以通过它们查询．

DATE、TIME 和 TIMESTAMP 类型也可以使用．前面两者是后面的子集．它的值由年、月、日、小时、分、秒和分秒组成．其形式（例如，JAN 25 1990,25 01 90）决定于各个工具(SQL／DS)或其他特殊形式(Oracle)．

只有 SQL／DS 支持数据类型 GRAPHIC、VARGRAPHIC 和 LOHG VARGRAPHIC．它们以双字节存贮．Oracle 支持 RAW 和 LOHG 数据类型．RAW 用于二进制数据．

2.2 定义表结构

数据定义语言（或 DDL）用来建立、修改和删除表结构．表结构由每个原组（字段）来定义．字段由名字、类型和另外的说明来定义．

定义表结构的一般文法是:

CREATE TABLE table-name

(field-name field-type [NOT NULL]

[{,field-name field-type [NOT NULL]}]…)

我们象下面那样来读上述语句．CREATE TABLE 是命令保留字．接下来是表名．然后是左括号，再接下来是第一个变量名和类型．选择项指定该域为 NOT NULL，意思是该字段不能是空的．缺省值为 NULL，意思是该字段可以是空的．

接下来的字段（是可选择的）可以用相同方法说明．用逗号来分割字段，右括号表示定义结束．SQL／DS 将表名和字段名限制为 14 个字符，符合 ANSI 标准．Oracle 允许 30 个字符．表里可以有 254 个字段．

图 2-1 和 2-2 分别为雇员表和定单表的定义．注，我们用大写字符表示命令、类型和选择项，名字用小写字符．

```
CREATE TABLE employees
(employee_no INTEGER NOT NULL,
first_name CHAR(20),
last_name CHAR(25),
street CHAR(30),
zip_code CHAR(5),
date_hired DATE,
type CHAR(1))
```

图 2-1 雇员表定义

在数据库中表名是唯一的，而字段名在一个表中要是唯一的．系统用用户 ID 标识码来指定表的创建者．这个信息是数据字典或系统目录的一部分（见第六章）．

我们指定某些字段为非空的．在定单表中，这避免用户进入一个没有定单号的、顾客

号、或收到定单日期的记录中．在雇员表中它避免用户进入无雇员号的记录．显然，表的主关键字必须是非空的，因为我们通过空字段是无法找到记录的．NOT NULL 是主关键字必要的但不是充分的条件，因为它的值必须是非一的和非空的．由于信息可能不存在或不可用，所以其它字段都可能是空的．

```
CREATE TABLE orders
(order_no INTEGER NOT NULL,
customer_no INTEGER NOT NULL,
employee_no INTEGER,
received DATE NOT NULL,
shipped DATE)
```

图 2-2 定单表的定义

删除表，使用命令：

DROP TABLE table-name

如前面所提的，关系数据库仅由表组成．一个表并不存在它自己本身的文件中．实际上，象 SQL／DS 和 Oracle 的数据库程序并不严格地规定每个数据库表的数目．(SQL／DS 限制为 8 百万个表!)．因此，由 100 个表组成的数据库可能只存在一个数据文件中．因此，数据库程序不能通过文件名来处理表．用户必须定义称为数据空间的单元并指定使用该空间的每个表定义．形式为：

CREATE TABLE test 768

(field definitions)

SPACE my-example-tables

如果不指定 SPACE，数据库系统将自动指定一个缺省值．数据空间的概念在关系数据库中与强调逻辑和物理数据库的独立性一样非常重要．

2.3 改变表的结构

有时由于新的需要我们必须修改表的结构．关系数据库管理系统使得某些改变（在实际上很平常）很容易．例如，我们可以方便地增加一个字段．删除或更改段的定义则较麻烦，但这并不经常需要．

在表中增加新的字段字义的命令如下：

ALTER TABLE table name

ADD field name type [{,field name type}]...例如，我们用下列命令将新的字段 middle-name 和 age 加入雇员表中：

ALTER TABLE employees

ADD middle name CHAR(11),age SMALLINT

Oracle 和 SQL／DS 允许用 ADD 字段．他们也将已存在的记录包含新的字段．通常我们不将新的字段指定为 NOT NULL，因为在已存在的记录中该段的值为 NULL 然而，在 Oracle 中只要表是空的，就可以指定非空并改变段的长度和数据类型．

处理非空表时，你必须完成中间步骤．建立一个临时表将数据放到该临时表中，再修改原始表(Oracle)或用新的定义(SQL／DS)来创立全新表，然后再将数据放回来．例如，在 Oracle 中，我们将雇员表中 first-name 一项增加到 30 个字符:

```
ALTER TABLE employees
MODIFY (first name CHAR(30) NOT NULL)
```

然而，必须小心．将返回值放到定义为 NOT NULL 的字段时可能出现问题，因为字串段的长度或字段类型已经改变．在 Oracle 中，可以将表名改变(RENAME old-table-name TO new-table-name)，但 SQL／DS 却不行．改变表的名字是有些冒险的，因此表名可以在数据库中的其它地方引用（在视图和应用中)．改变名字可能引起程序出错．

从定义中删除字段并不容易．问题是并不知道删除的字段在哪里引起问题，如在其它表中、程序中、或视图定义中引起问题．解决的方法见章节 3.3．

2.4 注释

作为表的创立者或 DBA 可以插入列或段的注释．ANSI 标准并不包含注释，但下列文法是 Oracle 和 SQL／DS 的注释文法:

```
COMMENT ON
{TABLE table-name |COLUMN table-name.column-name}
IS 'string'
```

使用下列命令可以在定单表中 order-no 字段上加注释:

```
COMMENT ON TABLE orders
IS 'the relationshpps among customers,orders,and employees'
```

或

```
COMMENT ON COLUMN orders order-no
IS 'unique determination of a particular order'
```

在表中和字段上的注释出现在系统目录中（见第六章)．

2.5 同名

表名经常很长（由于书写或语音的原故)，经常引用它们不方便也容易出错．如果使用别人的表那么问题就更严重，因为必须进入别人的表．解决的方法是引入简名或同名:

```
CREATE SYNONYM synonym
FOR owner-of-table table-name
```

只有执行这命令的人能使用同名．然而，它可用于由别人创立的表．如果 Jones 创立了表 delephone-list，你能用下列命令建立同名 tls:

```
CREATE SYNONYM tls
FOR jones telephone_list
```

你可以引用 tls 或 jones telephone_list

同名也记录在系统目录表中．注意，删除一个表并不能自动地删除同名，必须使用命令来删除它们:

DROP SYNONYM synonym

Oracle 中，DBA 可以将同名类型指定为 PUBLIC(CREATE PUBLIC SYNONYM synonym FOR table-name)所有用户能访问 PUBLIC 同名。