第一章　数据库管理的关系方法

Oracle 公司从推出自己的第一个数据库产品的那一天起，就声称它是"关系系统"。那么关系系统的含义究竟是什么呢？什么样的数据库产品才能称得上是关系系统？人们为什么都争先恐后地标榜自己的产品是关系型的？数据库管理的关系方法都有什么优点？所有这些问题就是本章所要讨论的主要内容。要想真正弄清这些问题，必须知道数据库管理系统的发展过程以及一些基本概念。本章第一节对数据库系统作一概括性回顾；第二节说明 Oracle 为什么是一个关系系统，并介绍关系系统的基本概念，对有较深厚的数据库知识的读者，可直接进入第二章的学习。

1.1　数据库管理为什么采用关系方法？

计算机问世之初，人们只是利用它处理复杂的科学计算工作。1954 年，通用电器公司(GEC)研制成功了第一台用于商业数据处理的电子计算机 UNIVAC1，计算机开始被用来进行事务管理，存储大量的信息文件。由此产生了电子信息系统。大量的日常事务数据可以有效地存储在当时的磁介质上，而且处理速度快得惊人，这解除了人们在大堆的文件夹中找数据之苦，因此基于计算机的信息处理系统迅速发展起来。

当时设计这类系统的方法是很专用化的，往往是根据具体的应用，建立一组文件，并编写一些处理这些文件的程序。但是当原有应用扩展变化之后，便要创建新的文件和编写新的程序。很快这种方法便暴露出以下的缺点：

- 数据冗余：系统设计完成后，发现很多文件其实都包含了同样的信息。例如：订货系统中所用到的数据可以同时为其它系统所使用，如库存管理，客户记帐等。由于各系统的数据是彼此独立存放的，相同的数据势必造成大量冗余存储。在一个大公司中，这可能意味着要浪费大量的存储空间。
- 数据不一致：上面提到的冗余不仅造成浪费，而且会带来数据潜在的不一致性。例如，若库存文件、订货文件、发票文件和顾客文件中都有货名这一项，由于文件是分离管理的，则对同一货物就有可能被赋于多个不同的名字，从而导致系统的不一致。
- 缺乏完整性和必要的控制，这也是数据冗余所带来的必然后果。不同文件重复存储相同信息，使得信息系统的全局管理和操纵变得相当困难，甚至是不可能的。对大多数企业，存储在文件中的这些数据就是维持企业运转的血液，然而传统的应用开发方法却会使得这一重要资源失去控制，成为散沙一片。

当人们认识到了原有方法所带来的这些弊病后，便开始采用一种新的、综合性的开发方法，即"数据库方法"。概括地讲，所谓数据库方法就是把一个系统所涉及的各种数据，综合整理为一组相互关联的文件，而不是按不同应用组织为各自独立的文件，使得数据共享。

图 1.1 就传统的方法和数据库方法作了比较，这里的应用背景是库存/订货管理。图 1.1 中的 1)是按传统方法组织数据的形式，应用中各相关信息被独立的分为三个文件，文件间的共享信息(如库存文件的价格和订货文件的单价，订货文件和顾客文件中的顾客名和地址)被冗余存放，这样数据的一致性就难以维护。如用户要修改库存文件中的价格时，必须同时修改订货文件中的单价，否则就会出现同一货物在系统中有两个价格的不一致情况。数据库方法与之不同，它是按图 1.1 中的 2)形式组织数据。这里同样分为三个文件，但此时的订货文件的一些数据是分别取自库存和顾客文件的。换句话说，库存和订货文件共享 StockNo、ItemName、Sprice，订货和顾客文件共享 CustomerNo、Name、Address。这样组织数据的好处是消除了数据冗余和数据的不一致。

数据库方法由于有以上优点，使得它在过去的二十多年里得到长足的发展，先后出现了层次方法、网状方法和关系方法，数据库方法是建立在数据模型之上的。所谓数据模型可以看作是一种形式化描述数据、数据之间的联系以及有关的语义约束规则的方法。上述三种方法就是分别建立在层次模型、网状模型和关系模型之上的，下面将简略地介绍一下这三种模型的基本含义和各自优劣所在。

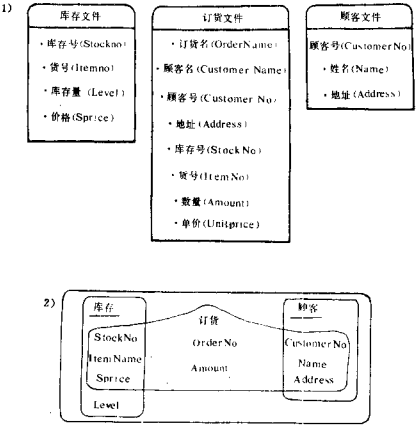

图 1.1 传统方法与数据库方法文件组织比较

1.1.1 层次数据库

1968年，IBM公司推出了信息管理系统(IMS)。这是较早地尝试采用数据库方法管理文件的系统。IMS即是层次数据库系统，它采用了层次模型作为组织数据的方式。

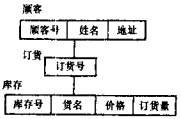

图 1.2(a) 顾客/订货/库存层次数据库模式

顾名思义，所谓层次模型，就是按照层次结构的形式来组织数据的数据模型。它把整个数据库的结构表示成为一个有序树的集合，而这些有序树的每一个结点是一个由若干数据项组合而成的记录型。

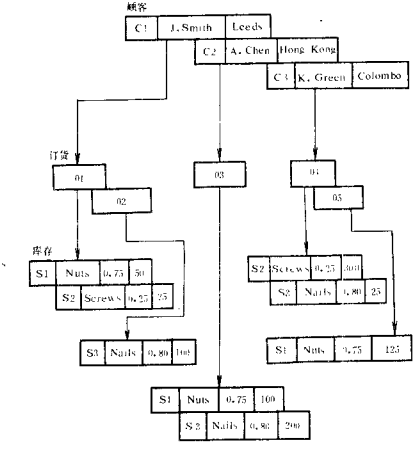

图 1.2(b) 顾客/订货/库存层次数据库实例

这里我们要解释一下数据库中常用到的型和值的概念，型是对客体的属性描述，是组织数据的一种框架；值则是指客体在型定义下的具体赋值。如顾客可定义为由编号、姓名、地址三个数据项组成的记录型，而(01，Smith，Leeds)则是这一记录型下的一个记录值。

层次模型的数据组织可归结为树的形式，因此它满足如下的两个条件：

1)整个模型只有一个根记录型；

2)根记录型以外的结点只有一个父结点,但任一记录型可以有零个或任意有限个子记录型。

在层次模型中,每一个有父子关系的记录型之间反映的是现实事物间的一个一对多联系,即父型的一个记录值可对应多个子型的记录值。

图 1.2(a)给出了采用层次模型组织顾客/订货/库存信息的型结构(称为数据库模式)。

这个层次体系结构的含义是:在库存/订货管理中,每一顾客可有多笔订单,每一订单中可订购多种库货物。

图 1.2(b)是在图 1.2(a)模式下的一个数据库实例。

用层次模型来描述具有较好层次关系的系统,会显得非常自然、直观。结构简单、层次清晰、易于理解是层次模型的突出优点。

但由于受层次结构的限制,层次模型还存在一些问题,比如就图 1.2(b)而言,库存记录 S1,S3 重复存三次,从而带来数据在某一层次上的冗余存放。造成这一情况的原因是层次模型的父子型之间只反映一对多联系,而现实中,订货/库存是多对多联系。另外层次模型有信息丢失的问题,比如对库存信息来说,只有被订购的库存货物才能被记录下来,非被订购的库存货物,由于没有父结点而无法存在于库中。最重要的问题是层次系统不能为用户提供灵活方便的完整性控制方法,故此人们又提出了新的数据模型——网状模型。

1.1.2 网状数据库

在七十年代,数据系统语言联合会(Conference On Data Systems Languages,简称 CODASYL)设立了一个数据库任务组(DBTG),来专门制定数据库实现准则。在 1971 年该组织正式公布了 DBTG 报告,对原有的层次模型做了修改和扩充,提出了新的模型,即 CODASYL 模型或网状模型。

网状模型的数据结构是从而不是树,和树结构一样,从结构也可以用父子结点来描述。但是,从结构允许任一结点无父结点,或有一个以上的父结点,从而构成一个复杂的网络结构。

图 1.3(a)是顾客/订货/库存的网状模型。

在网状模型下,记录型之间的联系是靠系型实现的。系型一般由一组非空的记录型命名集组成。在这一组记录型中,只有一个记录型处于突出位置,称为主记录型,而其余处于从属地位记录型称为属记录型。一个系型所反映的就是主记录型到属记录型的一对多的联系。如在 1.3(a)中,有三个系型 C-O、O-A 和 S-A,它们分别定义为:

C-O:(顾客;订货)

O-A:(订货;订货量)

S-A:(库存;订货量)

它的语义解释是:一个顾客可有多笔订货(C-O),一笔订货可订购多种货物(O-A),而一种库存货物可被多家订购(S-A)。

图 1.3(b)给出了网状模型下的顾客/订货/库存数据库实例。

与层次模型相比,网状模型的表示能力、精巧性和直接性都要强得多。网状模型不仅可以直接表示层次体系的系统结构,而且在表示网状结构体系,多元联系等方面都较层次模型直观、易于理解。

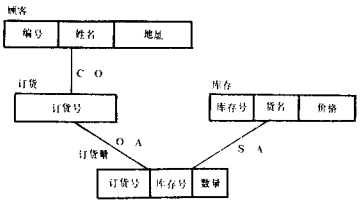

图 1.3(a) 顾客/订货/库存的网状模型

网状模型的严重缺点是其数据操纵的复杂性，用户在进行查询时，要按复杂的导航方式指明路径。

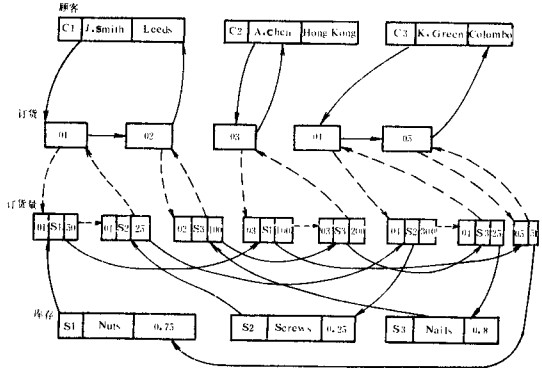

图 1.3(b) 顾客/订货/库存的网状数据库实例

网状模型和层次模型的共同缺点就是数据独立性较差，所谓数据独立性是指用户的应用不依赖数据存储的变化而变化。由于上述两种模型都过多地包含了许多物理存储细节，虽然它便得查询效率得以提高，但系统的数据独立性却降低了。

1.1.3 关系数据库

在 1970 年，美国 IBM 公司的研究员 E. F. Codd 发表了题为“大型共享系统的关系数据

模型"的论文，成为数据库领域中划时代的文献。文中首次提出了数据库系统的关系模型，该模型与以往的数据模型截然不同，已为数据库界所广泛地接受。

在关系模型中，文件被看作是两维表，它是由行和列组成的。严格地说，表应称为关系，行称为元组，列称为属性。这里的元组大致相当于文件中的记录，元组中的每一个值的含义由属性指明。前面的例子若用关系模型则表示为图 1.4 的形式

CUSTOMERS(顾客)

CUSTNO	CUSTNAME	ADDRESS
C1	J. Smith	Leeds
C2	A. Chen	Hong Kong
C3	K. Green	Colombo

STOCK(库存)

STOCKNO	SNAME	SPRICE
S1	Nuts	0.75
S2	Screws	0.25
S3	Nails	0.80

ORDERS(订货)

ORDERNO	CUSTNO
01	C1
02	C2
03	C2
04	C3
05	C3

ORDERLINES(订货量)

ORDERNO	STOCKNO	AMOUNT
01	S1	50
01	S2	25
02	S3	100
03	S1	100
03	S3	200
04	S2	300
04	S3	25
05	S1	125

图 1.4　顾客/订货/库存关系数据库

这里有四个关系，分别为 Customers、Orders、Stock 和 Orderlines。Customers 有三个属性(CustNO、CustName 和 Address)和三个元组，Stock 和 Orders 也是如此。Orderlines 表有三个属性，八个元组，每个元组反映了 Orders 与 Stock 之间的一个对应关系。各表中有重复属性，如 CustNO，OrderNo 和 StockNo，它们分别在表 Customers，Orders，Stock 中唯一标识一个元组。实体之间的联系是在这些公共属性上建立表而反映出来的。因此在关系方法中，实体和联系的表示是一致的。而在网状数据库中，联系是由专门的系结构所表示的。这样带来的好处是关系系统操作上统一、简单。比如在关系系统中要找出顾客 1 所订购的货物名称，就变得相当简单，首先在表 Orders 中找出顾客号为 1 的所有订购号，然后再用这些订购号在 Orderlines 表中找出对应的货物号，最后根据查得的货物号在表 Stock 中找出对应的货物名称。值得一提的是，在这整个过程中，用户不必关心查询路径，而只是给出所要做的事情即可。

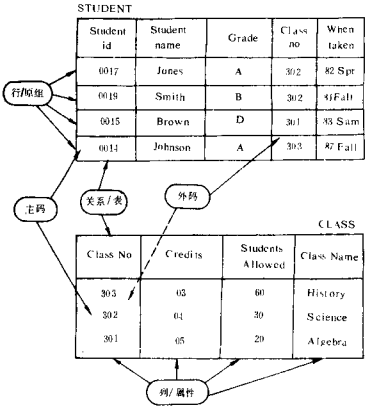

图 1.5　关系模型中主要概念示例

关系系统的成功在于它给用户提供了良好的开发环境，归纳起来主要有以下几方面优点：

1）有坚实的理论基础。关系数据库以集合论为基础，同时还研究和发展了关系数据库理论。

2）数据结构简单。实体和联系都用关系一体化表示，从而带来操作上的统一。而且操作是非过程化的，用户只需提出“干什么”而无需指出“怎么干”。因此操作简单，用户性能好。

3）具有高度的数据独立性。关系数据库把存取路径向用户完全隐蔽起来，用户存取数据时，既不考虑存贮结构也无需指出具体的存取路径。因而当数据的存贮结构或存取路径发生变化时不会影响应用程序的有效性。

4）面向集合的存取方式。与网状或层次模型中一次一个记录的存取方式不同，关系模型的存取方式是面向集合的。操作对象是一个或多个关系、操作结果也是一个新的关系。

由于关系系统具有以上这些优点，目前绝大多数的 DBMS 都在不同程度上基于关系模型。Oracle DBMS 杰出之处在于它是最早投放市场的商品化关系型 DBMS 之一。

要了解关系模型，除了关系、属性、元组等概念外，还需弄清以下几个重要的概念：

• 域：域是属性的取值范围，不同属性可取自同一个域。比如顾客号若是一个四位数字，则它所取自的域即是 0000～9999。

• 码、候选码、主码、外码：码是一个属性或一组属性，它能在关系中唯一地标识一个元组。CoustNo 是 Customers 表的码，StockNo 是 Stock 表的码，而 OrderNo 是 Orders 表的码，表 OrderJines 的码是由 OrderNo，StockNo 两属性组成的。一个关系上可能存在多个码。比如对学生关系，学号可作为码，身份证号也可作为码，在这种情况下，这些码称为候选码。实际被选中来充当码的候选码称为主码。某关系中的一个属性在其它关系中充当主码，则此属性称为本关系中的外码。例如，在 Orders 中，CustNO 即为一外码。在一个设计比较合理的关系数据库中，数据只在外码属性上是重复的，通过这种重复来建立关系间的联系。这样能简单灵活地保持数据的完整性，这要比网状和层次优越得多。为便于理解，现给出图 1.5 来示例上面讲到的各种概念。

1.2 Oracle DBMS 是关系系统吗？

关系方法区别于网状和层次方法的一个方面就在于它以严密的数学原理为基础，这样就有可能从理论上来确切地衡量一个 DBMS 是否是关系型的，这一点是极有用处的。因为目前许多厂家都声称自己的 DBMS 是关系型的，但事实上它们只支持很小一部分或根本就不支持 Codd 所提出的关系模型。目前，还没有一个产品能完全符合 Date(1986，1987，1988) 所定义的关系模型。但是已有一些产品(包括 ORACLE 系统在内)基本上支持关系模型的最主要特征。

按照 Date 的定义，关系模型有三个主要部分：

• 数据结构

• 关系操作

• 数据完整性

所谓关系系统就是支持上述关系模型的系统，但这种刻化是不精确的。下面分别介绍关系模型三个部分的主要含义，在此基础上给出关系系统的确切含义。

1.2.1 关系数据结构

上面已经提到，关系数据库所处理的是由行列组成的二维表。表的列称为属性，而行称为元组。表(或关系)的一个重要特征就是在行列的每一交点上的数据项都是原子的，即它是不能再分割为子集的项。例如：考虑 Orderlines 关系，若表示为图 1.6 的形式，则违背了属性值必须是原子的准则。关系模型中要求关系必须表示为图 1.4 的形式，关系中的每一单元(行列交点)只能容纳一个值，而不能是集合项或数组项。保持原子性所带来的一个好处就是只须为关系系统定义一个很小但功能很强的操作集就能满足 DBMS 所有标准的数据存取要求。

域是相同数据类型值的集合，例如顾客号就是所有合法顾客者号的集合，订货量域就是所有大于 0，小于(比如说)十万的整数集合，因此域是合法值集。出现在属性中的实际值就是从域中抽取的。域的意义如下：若两个属性从相同的域中抽取数据值，则涉及这两个属性的比较、连接和并等运算才会有意义。域就其性质而言基本上是概念性的，域是否要象实际

值集合那样明确地存储在数据库中无关紧要，但它们应作为数据库定义的一部分加以说明，其实目前大多数系统并不支持域的概念。

Order No	Stocknos	Amounts
1	{1,2}	{50,25}
2	{3}	{100}
3	{1,3}	{100,200}
4	{1,2}	{300,25}
5	{1}	{125}

图 1.6　Orderlines 文件带集合值的表示

1.2.2　关系数据操作

任何 DBMS 都要求有一组数据操作来从系统中存取数据。关系模型中只须有八个基本操作即能组合出所有的操作要求，它的总称为关系代数，所有这些操作都是从一个或两个关系中取得元组作为操作数并产生一组新的元组，构成一新的关系(或临时关系)。不要把关系代数和 Oracle 中使用的 SQL 语言相混淆，虽然两种语言在功能上是等价的，但所依据的基本概念不同。

Codd 最先定义了八个运算符，大致可分为两类。(1)基本运算：选择、投影、并、差和卡氏积；(2)可由上述导出的运算：连接、交和除。比如连接可由卡氏积、选择、投影来完成。图1.7给出了八个运算的示意图。下面分别介绍各运算。

1. 选择运算

选择运算是指从某关系中抽取满足一定条件的元组，它是对关系行的操作。例：对图1.4中的关系 Orderlines(注：以下所介绍的操作都以图 1.4 中的关系为例)可有如下的选择：

$$\triangle_{\text{stockno}=3}(\text{Orderlines})$$

其结果如下：

OrderNo	StockNo	Amount
2	3	100
3	3	200
4	3	25

2. 投影运算

投影运算产生关系的“垂直”子集，即通过选择指定的属性集上的值，然后删掉在所选择的属性中多余的重复值而得到的子集。例

$$\Pi_{\text{stockno}}(\text{Orderlines})$$

即把关系 Orderlines 在 StockNo 属性作投影，结果如下：

Stock No
1
2
3

3. 连接运算

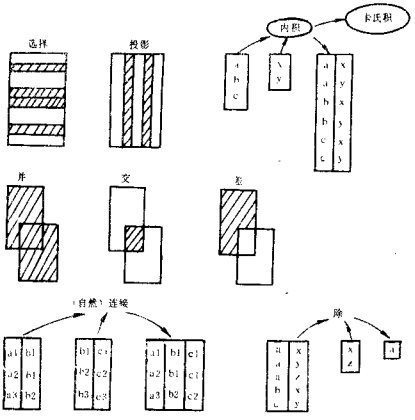

图 1.7　关系代数的八种运算

连接运算用于把两个关系合并为一个关系。最简单的连接条件是两个属性上相等的连接。当然还可以是大于、小于、不等的连接，连接时即是从两个表中分别各取一元组，并使之满足连接条件，直至所有元连接过为止。例：

$$\text{Orders} \underset{\text{OrderNo}}{\bowtie} \text{Orderlines}$$

意即关系 Orders、Orderlines 在属性 OrderNO 上作等值连接。结果见表 1-1。

4. 并运算

两个关系的并是由至少属于其一的所有元组组成。例：若关系 Customers 按顾客的住址划分为四个关系 NorthWest、SouthWest、NorthEast 和 SouthEast。则

$$(\text{NorthEast}) \cup (\text{SouthEast})$$

产生一个关系，它包含了所有居住在北部的客户。

5. 交运算

两个关系的交由同时属于两个关系的元组组成。例：

$$(\text{NorthWest}) \cap (\text{NorthEast})$$

产生一关系，其中每一客户都是身居两地的，既在西北部，也在东北部。

表 1.1 关系 Orders 和 Orderlines 连接的结果

OrderNO	CustNo	StockNo	Amount
1	1	1	50
1	1	2	25
2	1	3	100
3	2	1	100
3	2	3	200
4	3	2	300
4	3	3	25
5	3	1	125

6. 差运算

关系 A 和 B 的差是由属于 A 而不属于 B 的所有元组组成。例：

(NorthWest) − (NorthEast)

产生一关系，其中客户是住在西北部而不在东北部的。

交运算完全可由并和差运算来实现。例：

(NorthWest) ∩ (NorthEast)

= ((NorthWest) − ((NorthWest) − (NorthEast))) ∪ ((NorthEast) − ((NorthEast) − (NorthWest)))

但单独提供交运算会更方便些。

7. 卡氏积运算

该运算产生两个关系的笛卡尔积。例：

(Stock) × (Customers)

将产生表 1-2 的结果。它是由 Customers 中所有记录与 stock 中的所有记录相互配对而成的，连接运算其实是卡氏积的特殊形式。

表 1-2 Stock 与 Customers 卡氏积的结果

StockNo	Sname	Sprice	Custno	Custname	Address
1	Nuts	0.75	1	J. Smith	Leeds
2	Screws	0.25	1	J. Smith	Leeds
3	Nails	0.80	1	J. Smith	Leeds
1	Nuts	0.75	2	A. chan	Hong Kong
2	Screws	0.25	2	A. chan	Hong Kong
3	Nails	0.80	2	A. chan	Hong Kong
1	Nuts	0.25	3	K. Green	Colombo
2	Screws	0.25	3	K. Green	Colombo
3	Nails	0.80	3	K. Green	Colombo

8. 除运算

最容易说明除运算的是这样两个关系的除，其中关系 A 有两个属性，而关系 B 只有一个属性（见表 1-3）。A ÷ B 将产生结果如下：

Cust No

3

5

表 1-3 说明除法运算的关系 A,B

A		B
CustNo	StockNo	StockNo
1	1	1
1	2	2
2	2	3
3	1	
3	3	
3	2	
4	2	
4	3	
5	3	
5	2	
5	1	

这是因为,在关系 A 中,CustNo 为 3、5 的元组都能对应关系 B 中的所有 StockNo 值,即两关系的除运算返回的是被除关系中不与除关系相重的属性值,并且每一个这样的属性值在被除关系中所对应的元组都完全对应除关系中与之相重属性的所有值。

1.2.3 关系数据完整性

关系模型的三类完整性是实体完整性、参照完整性和用户完整性。

实体完整性和参照完整性是关系模型必须满足的完整性约束条件。应由关系系统自动支持。

· 实体完整性

实体完整性是基于主码的,一个主码可由一个或多个属性组成。实体完整性要求主码中的任一属性(即主属性)不能为空值,所谓空值是"不知道"或"无意义"的值。之所以要保证实体完整性主要是因为:在关系中,每一个元组的区分是依据主码值的不同,即每一个元组是靠主码值来标记的,若主码取空值,即不能标明该元组的存在。因此必须保证主码不为空。

· 参照完整性

参照完整性是基于外码的,若基本关系 R 中含有与另一个基本关系 S 的主码 KS 相对应的属性组 FK(FK 称为 R 的外码),则参照完整性要求,对 R 中每个元组在 FK 上的值必须为 S 中某个元组的 PK 值,或者为空值。

参照完整性的合理性在于:R 中的外码只能是对 S 中主码的引用,不能是 S 中主码没有的值。例 Class 和 Student 两关系,Class 中的学号是外码,它是 Student 的主码。若 Class 中出现了某个 Student 中没有的学号,即某个学生还没注册,却已有了选课记录,这显然是不合理的,但是外码可接受空值。例:在某公司中某雇员目前还未分配工作部门是可能的,这时该雇员的部门号属性(外码)则必须为空。

1.2.4 Oracle 五版是关系完备的关系系统

以上我们讨论关系模型的主要内容，归纳起来如下：

数据结构 S：

域(值)

n 元关系(属性，元组)

码(主码，候选码，外码)

数据完整性 I：

- 实体完整性：主码属性值不能为空
- 参照完整性：外码值必须与主码值匹配(或为空)

数据操纵 M：

关系代数

并、交、差、卡氏积

选择、投影、连接、除法

关系赋值

一个系统可称为是关系系统，至少要支持如下两点：

1)关系数据库。即在用户看来，数据库是由表构成的，并且只有表结构。

2)至少支持关系代数的选择、投影和连接运算，对这些运算不必要求定义任何物理存取路径。

一个系统仅支持关系数据库而没有选择、投影、连接运算功能，不能称为关系系统。一个系统虽支持这三种运算，但要求定义物理路径，也不能称为关系系统。当然并不要求关系系统明确支持选择、投影和连接运算符，而只要求有等价的这三种功能即可。

可见上面所定义的关系系统只是对关系模型的最小支持，它对关系模型的许多内容还不具备，故称这种关系系统是最小关系系统。根据系统对关系模型支持程度的不同，还可划分出多种关系系统。首先说明一下记号，一个关系系统记为 RS:=｛⟨S;s⟩⟨M;m⟩⟨I;i⟩｝，其中⟨S;s⟩、⟨M;m⟩和⟨I;i⟩分别表示系统 RS 所支持的数据结构、操作和完整性的内容。我们有如下各类的关系系统定义：

表式系统=｛⟨S;表/关系⟩｝

最小关系系统=｛⟨S;表/关系⟩⟨M;选择、投影、连接⟩｝

关系上完备系统=｛⟨S;表/关系⟩⟨M;八个运算⟩｝

关系上强完备系统=｛⟨S;表，码⟩⟨M;八个运算⟩⟨I;参照完整性，实体完整性⟩｝

全关系系统=｛⟨S;表，码，域⟩⟨M;八个运算⟩⟨I;参照完整性，实体完整性，用户定义完整性⟩｝

倒排文件系统属于表式系统；dbase、R；base 等属于最小关系系统；Oracle V5.1 以及 DB2、SQL/DS、INGRES 等是关系完备系统；Oracle V6.0 是关系强完备系统，目前还没有一个系统可称得上是全关系系统。

Oracle V5.1 作为关系上完备的系统理由如下：

1) Oracle V5.1 明显支持表结构，无论是用户数据还是系统数据(数据字典)都是以表存放的，但它没有码的概念。Oracle 只是提供予定义好的域，如整数型、数字型、字符型等，但没有

提供系统用户定义自己的域的手段。

2）Oracle V5.1 采用 SQL 语言。SQL 语句完全覆盖了前面所定义的八种运算，而且还扩展了集函数等功能，前面所示例的八种运算的 SQL 语言形式如下：

选择

```
SELECT    *
FROM      Orderlines
WHERE     StockNo=3
```

投影

```
SELECT   StockNo
FROM     Orderlines
```

连接

```
SELECT    Orders. * ,Orderlines. *
FROM      Orders,Orderlines
WHERE     Orders. OrderNo=Orderlines. OrderNo
```

并

```
SELECT * FROM NorthWest
UNION
SELECT * FROM NorthEast
```

交

```
SELECT * FROM NorthWest
INTERSECT
SELECT * FROM NorthEast
```

或写为

```
SELECT * FROM NorthWest
WHERE EXISTS
  (SELECT CustNo FROM NorthEast
  WHERE NorthEast. CustNo=NorthWest. CustNo)
```

差

```
SELECT * FORM NorthWest
MINUS
SELECT * FROM NorthEast
```

或写为

```
SELECT * FORM NorthWest
WHERE NOT EXISTS
        (SELECT CustNo FROM NorthEast
        WHERE NorthEast. CustNo=NorthWest. CustNo)
```

卡氏积

```
SELECT Stock. * ,Customers. *
FROM Stock,Customers
```

除法

```
SELECT DISTINCT CustNo
FROM A A1
WHERE NOT EXISTS
  (SELECT StockNo FROM B
    WHERE NOT EXISTS
      (SELECT * FROM A A2
      WHERE A2.CustNo=A1.CustNo
      AND A2.StockNo=B.StockNo)
```

3) Oracle V5.1 不支持参照完整性和实体完整性，SQL 语句没有定义和识别码的功能。虽然在 Oracle 中可通过定义某些属性不允许空值和唯一索引来实现实体完整性，但对关系的任一属性同样可定义为是允许空值的。这种保持完整性的方式是间接取得的，而不是系统自动保持的。因此在 Oracle 中有可能建立有重复元组的关系。而有重复元组的关系根本就不能称得上是严格意义下的关系，只不过是满足数据原子性的表。基于此原因，在本书后面的讲述中将不再称 Oracle 中数据结构为关系，而只称其为表。

综上所述，Oracle 是一个关系完备的系统，即它提供了基本的关系数据结构和操作。虽然完整性规则也是人们所期望的，但不是最根本的。要实现完整性规则会增加软件的开发复杂度和软件规模，却不会相应增加系统的功能。所以当时 Oracle 公司把目标定在增强用户功能上，即提供丰富的开发环境，而不是提供一个全关系系统。但完整性准则毕竟是衡量关系系统的重要标志之一，所以 Oracle 公司在其第六版中完全支持了参照完整性和实体完整性。用户可以直接定义主码、外码、参照完整性、实体完整性及用户完整性。但在 Oracle V6.0 中仍没有域的概念，所以它是关系强完备系统。

本章要点

- 数据库方法不仅能消除数据的冗余和不一致性，而且提供了对数据的集中控制和完整性控制的手段。
- 在三种数据库模型中，关系模型是唯一兼具灵活性和简洁性的模型。
- 关系模型基于数学理论基础，因此可以严格验证一个产品是否是关系的。
- 关系系统是支持关系模型的系统，依据支持的程度，可划分为多种类型。
- Oracle DBMS V5.1 可被认为是关系完备的系统，即它提供了关系模型所定义的基本数据结构和全部操作。

练习题

1.1 解释数据库一词的含义。

1.2 比较层次、网状、关系三种方法的优缺点。

1.3 解释下列名词：

关系，元组，属性，候选码，主码，外码，参照完整性，实体完整性，关系模型，关系系统。

1.4 参看图 1.4 中的关系，给出以下操作的结果：

a) $\triangle_{Address='Leeds'}$ (Customers)

b) Π Sname (Stock)

c)Stock  Orders

1.5 请说明以下各系统之间的异同：

a)最小关系系统

b)关系上完备系统

c)全关系系统

1.6 为什么说 Oracle DBMS 是关系上完备的系统？

第二章　Oracle 系统概述

Oracle 公司是美国一家著名的专门从事数据库技术研究、开发的计算机软件生产厂家，它创立于 1977 年，自创建以来的十多年中，先后推出多种版本的 Oracle 产品，1986 年推出的 Oracle V5.1 是一个具有分布处理功能的关系系统，1988 年底推出的第六版则向全关系系统迈进一步，同时加强了事务处理的功能，1991 年推出了 Oracle V6.2，旨在使事务处理上有突破性进展。

目前 ORACLE 产品覆盖了大、中、小型机、微机等 60 多种机型，成为世界上使用最广泛的性能最优的数据库管理系统。Oracle 一进入中国市场，就迅速为中国用户所接受，并且美国 Oracle(中国)有限公司在与有关单位合作下推出了 5.1 版的汉化产品，使得所有数据、表名、列名都可使用汉字。

Oracle 能在短时间内获得如此巨大的成功，关键在于它有如下的技术特点：

· 可兼容性：Oracle 采用 SQL 作为数据库查询语言，由于 SQL 已成为美国国家标准，所以用户开发的 Oracle 应用软件可在 IBM 的 DB2 和其它基于 SQL 的数据库上运行。

· 可移植性：Oracle 拥有 60 多种大中小型机、工作站及微机上的运行版本，其中包括 IBM、DEC、HP、SUN 等著名厂家的计算机都配有相应的 Oracle 产品，并且 Oracle 可以运行在 UNIX、VMS、XENIX、DS12、DOS 等多种操作系统下，因此可以把 Oracle 作为标准的数据库管理系统，当硬件配置变化时，无须做任何改动，原来设计的 Oracle 应用可照搬使用。

· 可连接性：Oracle 采用分布式共享方式，从而使 Oracle 数据库能够驻留在多台计算机之上，加之 Oracle 可运行在不同的机种及不同的操作系统下，从而可以将不同的计算机，操作系统联成网络，达到异型机联网、数据分布共享的目的。

· 多功能性：Oracle 的优越性体现在其先进的体系结构即第四代应用开发环境上，它将数据库应用分成前端和后端处理两部分，这样就保证了 Oracle 具有最大的吞吐量、多用户事务处理、保护数据以免非法存取及系统失败的能力，此外 Oracle 公司还为各种层次的用户提供了一系列的应用开发工具，从而增强了设计能力。

· 高生产率：在 Oracle 所提供的第四代开发环境下从事应用程序开发，效率是极高的，开发人员可以采用“原型法”直接在 SQL * Form 上进行设计，在用户的参与下，渐次地完成最终的用户要求。

从本章开始将展开对 Oracle 的各系统层面和应用层面的介绍，所基于的是 Oracle V5.1，本章第一节简述 Oracle 的软件总体结构，第二节介绍 Oracle 的进程结构，第三节综述 Oracle 的产品结构。