数据库丛书

数 据 模 型

刘惟一 田 雯 编著

科学出版社

2001

内 容 简 介

本书全面地介绍了数据模型设计的理论和建模方法，其重点是数据库的逻辑模型，同时也涉及到概念模型。本书共分六章，第一章介绍了关系数据理论；第二章讨论了模糊关系数据理论；第三章将面向对象方法和模糊语义融于建模中，讨论了模糊数据模型；第四章讨论了时态数据模型；第五章介绍概率关系模型；第六章着重介绍了多媒体和信息检索及其它数据模型。

本书尽可能地提出一些具体问题作为抽象建模的引导，对所讨论的问题几乎都给出了相当详细的证明和应用实例，便于读者理解和运用。

本书可作为计算机、信息系统等专业的研究生及高年级本科生的教材，也可供从事数据库、人工智能和信息管理工作的科技、工程人员使用和参考。

图书在版编目(CIP)数据

数据模型/刘惟一等编著. -北京：科学出版社，2001
(数据库丛书)
ISBN 7-03-005860-7

Ⅰ. 数… Ⅱ. 刘… Ⅲ. 数据模型 Ⅳ. TP311.13

中国版本图书馆 CIP 数据核字(2001)第 028208 号

科学出版社 出版
北京东黄城根北街 16 号
邮政编码：100717
新蕾印刷厂 印刷
科学出版社发行 各地新华书店经销

*

2001 年 6 月第 一 版 开本：787 × 1092 1/16
2001 年 6 月第一次印刷 印张：17 3/4
印数：1—3 000 字数：40 000

定价：28.00 元

(如有印装质量问题，我社负责调换〈环伟〉)

前　言

数据库技术是计算机科学技术领域中发展最快、应用最广的技术之一。谈到数据库技术不能不讨论数据库设计，谈到数据库设计又不能不讨论数据库模型。本书的重点是研究数据库的逻辑模型，但在讨论中不可避免地要涉及到概念模型，将这两方面内容统一起来称为“数据模型”。在研究数据模型时，既要讨论它的设计理论同时还要讨论它的设计方法。进一步说，本书讨论的是数据模型设计的理论和方法。

华罗庚先生曾把数学研究比作下棋(《数论导引》)，他说：必须多和“高手”下棋(即把数学大家的结果试予改进)，必须多揣摩成局(著名问题的证明)。因此本书在试图概述这一领域的最基本、最重大的成果时，也介绍这一领域的最新研究进展以及作者在这一领域中的工作。

从具体到抽象是数据库建模的主要方法，具体例子往往是抽象模型的思想源泉，因此本书尽可能地提出一些具体问题作为抽象建模的引导，以便给读者一些感性认识。本书对所讨论的问题几乎都给出了相当详细的证明和应用实例。本书没有讨论具体的实现细节，因为实现方法的讨论往往陷入问题的枝节而使解决问题的思想模糊。虽然由算法和示例过渡到实现并不总是十分容易的，但没有实质性困难。

对不同的应用领域，需要有不同目的的设计模型，当然也就有不同的理论和方法。本书从不同角度就以下几个方面的内容展开了讨论。第一章介绍关系数据理论，它是以后章节的基础，其主要内容在 Ullman 的《数据库系统原理》及其它教科书中都有论述。和 Ullman 的教材相比，本章有以下几个特点：把无损连接看成是一种数据依赖，强调了依赖理论的和谐性和系统性；引入追逐表法对蕴涵问题进行检验，专门讨论了数据依赖蕴涵。第二章讨论模糊关系理论。在这一章人们可以看到将关系理论扩展到模糊环境的方法，以及模糊数据依赖的应用。这里作者建立了一套将精确值、模糊值、空值统一起来的数据依赖系统。第三章讨论模糊数据模型。这一章将面向对象方法和模糊语义溶于建模中，主要讨论了模糊语义关联、模糊继承层次的划分、带概括语义和聚集语义的模糊数据模型以及数据模型的可满足性判定。特别是将属性间依赖关系扩展为模式间的依赖关系实现表的继承，所提出的方法，在进行表的层次链接时都可以借鉴。第四章讨论时态数据模型。Gregesen，Tansel 和 Jensen 等人分别对时态概念模型、时态逻辑模型和时态数据依赖及范式作了全面综述。本章的相关内容取材于他们的论文。作者在这一章提出了稳定约束的概念，给出了解决带时滞的约束和关联问题的方法。第五章讨论概率关系模型。概率依赖关系是知识表示的一种重要形式，通过图形来描述概率依赖关系是一种直观有效的方法，马尔可夫网和贝叶斯网就是最常用的两种概率图示，关于这方面的内容本书主要取材于 Pearl 的著作。这一章还介绍了 Wong 等人关于扩张概率关系模型及概率数据依赖方面的工作。最后介绍由数据依赖构造概率网络的方法，这是作者的工作成果。第六章对一些新近提出的数据模型作了简要介绍。数据库技术总是在整个信息技术发展的大背景下发展的，海量信息、网络化、多媒体化对数据库技术提

出了新课题，相应地产生了多媒体模型和信息检索模型。这一章主要介绍了多媒体文件的语义数据模型、多媒体对象的时间、空间约束和文本分类检索方面的内容。多媒体系统的复杂性不仅由于它的对象是文本、图形、图像、声音等，更主要的是这些多媒体数据与时间、空间约束紧密相连，对象的持续时间及他们的时态关系都要受到严格的约束。这一章借助 Petri 网方法描述多媒体时间、空间约束模型，同时也对 Petri 网模型作了简要介绍。另外，就文本分类检索问题进一步介绍了基于概率推理的信息检索建模方法。

在成书过程中得到中国人民大学王珊教授的鼓励和支持，香港城市大学廖少毅博士和王槐清博士为作者提供了方便的研究环境和条件。云南大学研究生梅伟、何盈捷同志为本书制作了大量图表。另外本书还得到国家自然科学基金(编号 69763003)的资助，在此一并表示衷心的感谢。

由于作者水平有限，书中错漏和不妥之处在所难免，恳请读者批评指正。

作　者

2001 年 1 月

第一章　关系数据理论

1.1　关系数据库概述

数据库技术发展至今已有近30年的历史，它作为数据管理的有效手段，大大促进了计算机应用技术的发展。发展这一技术，使之更加系统，有效已成为数据库研究的主要方向。

从早期的文件系统到层次和网状数据库，从关系数据库到面向对象数据库，以及面向不同应用的时态数据库，演绎数据库等均已向人们展现了数据库技术的广阔的应用前景。

关系型数据库由于其自身的直观性与建立于其上的理论的系统性，从诞生起便得到了人们的广泛关注，在很大范围内得到了应用上的推广，直至今日，它仍是一种常用的数据模型。从本节开始，我们将讨论关系数据库理论以及它的应用，这是本书的基础。

实际上，关系模型的逻辑结构就是一个二维表格。学生情况的关系表如表1.1.1所示。

表格的第一行是各种属性名，以下每一行表示了某一学生的信息。为方便问题的讨论，我们引入下列基本术语与概念。

表1.1.1　学生情况表

学号	姓名	性别	年龄
96001	Li	男	20
96002	Mou	女	18
96009	Zhang	女	19

1. 术语

1）属性　如表1.1.1所示，表中的一列作为属性，属性有属性名和其所对应的值域。属性的值为值域中一个具体元素。表1.1.1中，“性别”是该列的属性名，其值域为{男，女}，而一个属性值是某一行上一个具体的取值{男}或{女}。

2）关系模式　是二维表的框架，为属性名的集合，即表格的表头各项的集合。在图1.1.1中，关系模式为{学号，姓名，性别，年龄}。

3）关系实例　是二维表中各行具体取值的集合，它是客观事物在此表格各属性的值的表现，是具体的实例。表1.1.1中，关系实例为：(96001，Li，男，20)，(96002，Mou，女，18)，……

4）元组　是实例中的一行，是某个现实世界事物在此关系模式下值的表现，反映了该事物的某些特征，在表1.1.1中，(96001，Li，男，20)是一个元组。由此，所有表中元组是关系实例中具体的一个实例。

2. 定义

上面所阐述的是直观的描述，为使今后讨论更加方便，我们引入形式化定义：

定义1.1.1　给定一组属性的值域为$D_1,D_2,\cdots,D_n$。它们的笛卡儿积$D_1\times D_2\times\cdots\times D_n=\{(d_1,d_2,\cdots,d_n)\mid d_i\in D_i,i=1,2,\cdots,n\}$。其中，$(d_1,d_2,\cdots,d_n)$作为一个元组，某个$d_i$作为分量。关系实例是$D_1\times D_2\times\cdots\times D_n$的子集，也称为在域$D_1,D_2,\cdots,D_n$上的关系(relation)。

由此，可知关系是一张二维表，表的每行对应一个元组，表的每列对应一个属性，有具体

的名与值。

3. 关系的性质

我们需要对关系作出人为规定，以避免使用过程中产生错误与混乱。

1）关系表中每一分量（属性值）必须是不可分的数据项，例如不能是诸如“1973 年 8 月”。这样的值是可分的，故不可用于表中某一分量。

2）同一关系实例中任两个元组不能完全相同。

3）任一个关系模式中各属性名不能相同，否则无法区分。两个表若相关，同一属性名的属性值域应相同，这才使连接两张表成为可能。

4）行列的次序无关，即哪一行（列）在前或在后，不影响表的性质。

4. 说明

1）关系模式　关系的描述称为关系模式，它包含属性名，属性名向值域的映像，记为 $R(A_1, A_2, \cdots, A_n)$。对一个应用来说，模式是不变的，而变化的是模式上的实例。

2）关系数据库的实现

① 把每个关系二维表格当作一个文件来存储，记录的字段对应于表中的属性。

② 文件的组织可以是索引文件也可以是散列的文件。

以上，我们介绍了关系数据库模型的基础定义与概念，今后的讨论都将在其上展开。

1.2 关系代数

1.2.1 关系代数及其运算

关系代数分为传统的集合运算和特殊的关系操作两种，我们将分别予以介绍。

1. 符号

常用的关系代数运算符如下：

(1) 集合运算符

$\cup$：并；

$\cap$：交；

$-$：差；

$\times$：广义笛卡儿积。

这类运算将关系看作元组的集合，它的运算是从关系的“水平”方向，即行的角度来进行的。

(2) 特殊的操作符

Π：投影；

σ：选择；

$\bowtie$：连接；

$\div$：除。

这是在关系表的行和列中按用户需要而进行的运算。

(3) 比较符

$>$:大于;

$<$:小于;

$\neq$:不等于;

$\leqslant$:小于等于;

$\geqslant$:大于等于;

(4) 逻辑运算符

$\neg$:非

$\wedge$:与

$\vee$:或

2. 集合运算

在其中,$\cup$,$\cap$和$-$是在同一关系模式下两个实例之间的运算。

我们通过例子来说明。

例 1.2.1 关系实例 r 与 s,p 如表 1.2.1 所示。

表 1.2.1 关系实例

r

A	B	C
a	b	c
d	e	f
g	h	i

s

A	B	C
a	b	c
j	k	l

p

E	F
d	e
f	g

则这三个关系模式的集合运算结果如表 1.2.2 所示。

表 1.2.2 集合运算

$r\cup s$

A	B	C
a	b	c
d	e	f
g	h	i
j	k	l

$r-s$

A	B	C
d	e	f
g	h	i

$r\times p$

A	B	C	E	F
a	b	c	d	e
a	b	c	f	g
d	e	f	d	e
d	e	f	f	g
j	h	i	d	e
j	h	i	f	g

$r\cap s$

A	B	C
a	b	c

$s-r$

A	B	C
j	k	l

3. 特殊的关系操作

我们仍用例子来说明 Π,σ 和$\bowtie$的用法。

例 1.2.2 关系模式 S,C 表示学生模式与课程模式如表 1.2.3 所示。

表 1.2.3 学生课程表

S

$S^{\#}$(学号)	S_n(学名)	SD(所在系)	SA(年龄)
s_1	N_1	Computer	20
s_2	N_2	Maths	21
s_3	N_3	Maths	19
s_4	N_4	Computer	20

C

$C^{\#}$(课号)	C_n(课名)	$PC^{\#}$(先行课号)
c_1	P	—
c_2	D	c_3
c_3	O	c_1
c_4	D	c_3

表 1.2.4 学生选课表

SC

$S^{\#}$	$C^{\#}$	G(成绩)
s_1	c_1	A
s_1	c_2	A
s_1	c_3	B
s_2	c_1	B
s_2	c_3	C
s_3	c_4	A
s_3	c_2	B
s_4	c_1	B
s_4	c_4	C

而学生选课的二维关系表为 SC，如表 1.2.4 所示。

我们分别介绍上述几种运算：

(1) 选择(σ)

$$\sigma_F(r)=\{t \mid t\in r \wedge F(t)=\text{true}\}$$

其中，F 为逻辑公式。表达式意味着从关系实例 r 中选取适合条件 F 的元组集合。例如例 1.2.2 中，选取 $\sigma_{(SD=\text{'maths'} \wedge SA<\text{'20'})}$的元组，其结果为$<s_3,N_3,\text{Maths},19>$。

(2) 投影(Π)

$$\Pi_{A_i}(r)=\{t[A_i] \mid t\in r\}$$

其中，$t[A_i]$表示元组 t 中属性 Ai 的值。其含义为在关系实例 r 中选择若干属性列 Ai 来组成一个新的关系，例 1.2.2 中，$\Pi_{S^{\#},SD}(S)$和$\Pi_{SD,SA}(S)$如表 1.2.5 所示。

表 1.2.5 投影

$T_{s^{\#},SD}(S)$

$S^{\#}$	SD
s_1	Computer
s_2	Maths
s_3	Maths
s_4	Computer

$\Pi_{SD,SA}(S)$

元组号	SD	SA
t_1	Computer	20
t_2	Maths	21
t_3	Maths	19
t_4	Computer	20

由表 1.2.5 可见，上面第 t_1 与 t_4 元组完全一致，由于关系中也不能有相同元组(否则冗余)，故我们将 t_4 去掉。

(3) 连接($\bowtie$)

连接常分为两种：条件连接和自然连接。

条件连接

$$r \underset{A\theta B}{\bowtie} s=\sigma_{A\theta B}(r\times s)$$

其含义为：在 r 与 s 的笛卡儿积中选取那些满足条件 $A\theta B$ 的元组，其中 θ 为比较符。

例 1.2.3 关系模式 r 与 s 和 $r \underset{B>E}{\bowtie} s$ 如表 1.2.6(b)所示。

自然连接的含义为两个关系的共有属性的等值连接，$r\bowtie s=\{\overset{\frown}{t_r t_s}[\overline{B}] \mid t_r\in r, t_s\in s \wedge t_r[B]=t_s[B]\}$，其中$\overset{\frown}{t_r t_s}$表示两个 r 与 s 中元组的衔接，也称为联串，B 是 r 与 s 共有的属性。

例 1.2.4 针对例 1.2.2 中的关系实例，C$\bowtie$SC 的结果如表 1.2.7 所示。

表 1.2.6 连接

r

A	B	C
1	3	5
2	4	6

s

E	F
4	7
2	3
1	2

(a)

$r \underset{B>E}{\bowtie} s$

A	B	C	E	F
1	3	5	2	3
1	3	5	1	2
2	4	6	2	3
2	4	6	1	2

(b)

表 1.2.7 C,SC 连接表

$S^{\#}$	$C^{\#}$	G	Cn	$Pc^{\#}$
s_1	c_1	A	P	—
s_1	c_2	A	D	c_3
s_1	c_3	B	O	c_1
s_2	c_1	B	P	—
s_2	c_3	C	O	c_1
s_3	c_4	A	D	c_3
s_3	c_2	B	D	c_3
s_4	c_1	B	P	—
s_4	c_4	C	D	c_3

为了进一步熟悉掌握这些运算，我们用一些例子来说明它们的用法。

4. 例子

给定下面两组模式：一组为学生、课程和选课模式，另一组为教师、成果和排序模式。

学生模式 $S=\{S^{\#}(学号),S_n(学生名),S_D(所在系),S_A(年龄)\}$

课程模式 $C=\{C^{\#}(课号),C_n(课程名),Pc^{\#}(先行课号)\}$

学生选课模式 $SC=\{S^{\#}(学号),C^{\#}(课号),G(成绩)\}$。

教师模式 $T=\{T^{\#}(教师号),T_n(名字),T_t(职称)\}$

成果模式 $A=\{A^{\#}(成果号),A_n(成果名),A_i(时间)\}$

排序模式 $O=\{T^{\#},A^{\#},Or(排名)\}$。Or 指某教师 $T^{\#}$ 在某成果 $A^{\#}$ 中排名。

这里模式上的实例名与模式名不加区别。

例 1.2.5 求选 c_1 课的学生所在的系：

$$\Pi_{SD}[\Pi_{s^{\#}}(\sigma_{c^{\#}='c_1'}(SC))\bowtie S]$$

例 1.2.6 张三在 A_1 成果中的排名：

$$\Pi \text{ or } (\sigma_{A^{\#}='A_1'}(\Pi_{T^{\#}}(\sigma_{T_n='张三'}(T))\bowtie O))$$

例 1.2.7 选修过与 c_2 课为先行课的课程的学生的名字：

$$\Pi_{sn}\{\Pi_{s^{\#}}[\Pi_{c^{\#}}(\sigma_{pc^{\#}=(c_2)}C)\bowtie SC]\bowtie S\}$$

例 1.2.8 参加完成 A_1 及 A_2 成果的教师名单。

如果我们将其写为 $\sigma_{A^{\#}=A_1\wedge A^{\#}=A_2}(O)$是错误的，每一元组中 $A^{\#}$ 不可能为两个值，这里 $F(t)$=true 是对于一个元组 t 使其值为真。

正确的解法是 $\Pi_{Tn}\{[\Pi_{T^{\#}}(\sigma_{A^{\#}='A_1'}(O))\cap\Pi_{T^{\#}}(\sigma_{A^{\#}='A_2'}(O))]\bowtie T\}$

例 1.2.9 求 1982～1994 年之间的成果名称。

$$\Pi_{An}[\sigma_{Ti\geqslant 82}(A)]\cap\sigma_{An}[\sigma_{Ti\leqslant 94}(A)]$$

例 1.2.10 以张三为第一完成者的成果名。

$$\Pi_{An}[\Pi_{A^{\#}}[\sigma_{or=1}(\Pi_{T^{\#}}(\sigma_{Tn='张三'}(T))\bowtie O)]\bowtie A]$$

例 1.2.11 不选 c_1 课程的学生名。

$$\Pi_{Sn}[(\Pi_{s^{\#}}(S)-\Pi_{s^{\#}}(\sigma_{s^{\#}='c_1'}(SC))]\bowtie S]$$

例 1.2.12 选且仅选 c_1 和 c_3 的学生的学号。

我们分步骤来求解：

1）至少选 c_1，c_3 的学生的学号集 A：

$$A=\Pi_{s^\#}(\sigma_{c^\#}=\text{`}c_1\text{'}(\text{SC}))\cap\Pi_{s^\#}(\sigma_{c^\#}=\text{`}c_3\text{'}(\text{SC}))$$

2）除 c_1 与 c_3 之外的课程的课号集 B：

$$B=\Pi_{c^\#}(\text{C})-[\Pi_{c^\#}(\sigma_{c^\#}=\text{`}c_1\text{'}(\text{C}))\cup\Pi_{c^\#}(\sigma_{c^\#}=\text{`}c_3\text{'}(\text{C}))]$$

3）选了 B 集课程的学生的学号集 B^*：

$$B^*=\Pi_{s^\#}(B\rtimes\text{SC})\text{或 }\Pi_{s^\#}(\sigma_{c^\#}=\text{`}B\text{'}(\text{SC}))$$

4）所求结果为：$A-B^*$。

我们再介绍一下另一运算符÷。

给定关系 r 和 s 的模式为$<X,Y>$和$<Y,Z>$，其中，X,Y,Z 为属性集。

$$r\div s=\{t_r[X]\mid t_r\in r\wedge Y_x\supseteq\Pi y(s)\}$$

其中，Y_x 为 x 在 r 中的象集，而 $x=t_r[X]$。

我们用例子进一步说明。

例 1.2.13 在例 1.2.12 中，至少选 C_1，C_3 课程的学生的学号。

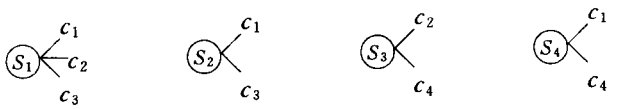
有两种方法，前一种为

$$\Pi_{s^\#}(\sigma_{c^\#}=\text{`}c_1\text{'}(\text{SC}))\cap\Pi_{s^\#}(\sigma_{c^\#}=\text{`}c_3\text{'}(\text{SC}))$$

这已在例 1.2.12 中介绍过。

后一种我们构造 T 为 $C^\#=\{c_1,c_3\}$，则 $\Pi_{s^\#c^\#}(\text{SC})\div T$ 为所求。

$\Pi_{s^\#c^\#}(\text{SC})$表示为



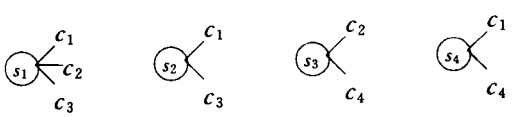
选 c_1，c_3 课的有 s_1 与 s_2，故我们得到这样的 s_1，s_2，它所对应的 $C^\#$ 中包含了我们构造的 T 中的课号 c_1 与 c_3。

例 1.2.14 选修所有课程的学生的学号。

方法一： $\Pi_{s^\#c^\#}(\text{SC})\div\Pi_{c^\#}(\text{C})$

以表 1.2.4 为例，$\Pi_{c^\#}(\text{C})=\{c_1,c_2,c_3,c_4\}$

$\Pi_{s^\#c^\#}(\text{SC})$可表为



除法得到的结果为空集。

除法不是基本运算，它可用下面的表达式表示。

设 r 和 s 的模式分别为$<XY>$和$<YZ>$。

$$r\div s\triangleq\Pi_X(r)-\Pi_X(\Pi_{XY}(\Pi_X(r)\times s)-r)$$

运用这个表达式我们得到方法二。

1）$T=\Pi_{S^\#}(\text{SC})=\{s_1,s_2,s_3,s_4\}$

2) $P=(T\times\Pi_{s^{\#}}(\mathrm{C}))-\Pi_{s^{\#}c^{\#}}(\mathrm{SC})=\begin{cases}s_1c_1 & s_1c_2 & s_1c_3 & s_1c_4\\ s_2c_1 & s_2c_2 & s_2c_3 & s_2c_4\\ s_3c_1 & s_3c_2 & s_3c_3 & s_3c_4\\ s_4c_1 & s_4c_2 & s_4c_3 & s_4c_4\end{cases}$

3) $\Pi_{s^{\#}}(P)=Q=\{s_1,s_2,s_3,s_4\}$

4) $T-Q=\varnothing$

1.2.2 查询优化

在数据库技术的应用中，我们常需要作查询的工作，因而，查询的效率就显得十分重要。这里，我们将讨论如何使查询得到优化。

我们先用一例子来引出讨论。

1. 引例

在 1.2.1 节的例子中，求选择 c_2 课的学生的名字。

我们可采用三种方法：

$Q_1=\Pi_{\mathrm{sn}}(\sigma_{\mathrm{sc}\cdot c^{\#}=\text{'}c_2\text{'}}\wedge_{\mathrm{sc}\cdot s^{\#}=s\cdot s^{\#}}(\mathrm{SC}\times\mathrm{S}))$ 10^5 秒

$Q_2=\Pi_{\mathrm{sn}}(\sigma_{\mathrm{sc}\cdot c^{\#}=\text{'}c_2\text{'}}(\mathrm{SC}\bowtie\mathrm{S}))$ 205 秒

$Q_3=\Pi_{\mathrm{sn}}(\mathrm{S}\bowtie(\sigma_{\mathrm{sc}\cdot c^{\#}=\text{'}c_2\text{'}}(SC))$ 10 秒

三种方法运行时间差异很大。

由此可见，虽然三种方法均可得到查询结果，但其效率有本质上的差别。因此，我们试图提供一些方法与策略来优化查询过程。

2. 常用的等价变换

我们先提供一些变换公式：

1) 连接满足交换率

$$E_1\times E_2=E_2\times E_1$$
$$E_1\bowtie E_2=E_2\bowtie E_1$$
$$E_1\underset{Q}{\bowtie}E_2=E_2\underset{Q}{\bowtie}E_1$$

2) 连接满足结合律

$$(E_1\times E_2)\times E_3=E_1\times(E_2\times E_3)$$
$$(E_1\bowtie E_2)\bowtie E_3=E_1\bowtie(E_2\bowtie E_3)$$
$$(E_1\underset{Q}{\bowtie}E_2)\underset{Q}{\bowtie}E_3=E_1\underset{Q}{\bowtie}(E_2\underset{Q}{\bowtie}E_3)$$

3) 投影串接

$$\Pi_A(\Pi_B(r))=\Pi_A(r)，\text{其中 } B\supseteq A$$

4) 选择串接

$$\sigma_{F_1\wedge F_2}(r)=\sigma_{F_1}(\sigma_{F_2}(r))$$

F_1,F_2 满足交换律。

5) 选择投影的交换律

$$\Pi_A(\sigma_F(r))=\sigma_F(\Pi_A(r))$$

6）选择与笛卡儿积的交换律

$$\sigma_{F_1}(r_1\times r_2)=\sigma_{F_1}(r_1)\times r_2 \quad (F_1 \text{只涉及} r_1)$$

$\sigma_{F_1\wedge F_2}(r_1\times r_2)=\sigma_{F_1}(r_1)\times\sigma_{F_2}(r_2)$ （F_1 涉及 r_1，F_2 涉及 r_2），

$\sigma_{F_1\wedge F_3}(r_1\times r_2)=\sigma_{F_3}(\sigma_{F_1}(r_1)\times r_2)$（$F_1$ 涉及 r_1，F_3 涉及 r_1，r_2）

7）选择与并交换，与交交换

$$\sigma_F(r_1\cup r_2)=\sigma_F(r_1)\cup\sigma_F(r_2)$$

$$\sigma_F(r_1\cap r_2)=\sigma_F(r_1)\cap\sigma_F(r_2)$$

8）选择与差交换

$$\sigma_F(r_1-r_2)=\sigma_F(r_1)-\sigma_F(r_2)$$

9）投影与笛卡儿积交换

$\Pi_{AB}(r_1\times r_2)=\Pi_A(r_1)\times\Pi_B(r_2)$，（$A$ 涉及 r_1，B 涉及 r_2）

10）投影与并交换

$$\Pi_A(r_1\cup r_2)=\Pi_A(r_1)\cup\Pi_A(r_2)$$

3. 优化算法

我们可以利用以上的变换公式使查询的关系表达式得以优化。下面，给出优化的具体算法。

我们先看一个引例。

在上一节中，求选 c_1 课程的学生姓名，其表达式的语法树为图 1.2.1(a)，图 1.2.1(a)可以化为图 1.2.1(b)。

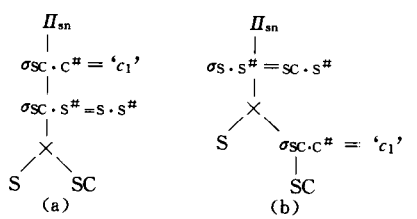


图 1.2.1 语法树

由上例可知，我们应尽早做投影和选择，以避免连接时的过多冗余，由此，得到下面算法。

算法：关系表达式的优化

输入：待优化的关系表达式语法树

输出：一组优化的顺序表达式

步骤 1 把 $\sigma_{F_1\wedge F_2\wedge\cdots F_n}(r)$ 按规则转化为 $\sigma_{F_1}(\sigma_{F_2}(\cdots(\sigma_{F_n}(r))\cdots)$；

步骤 2 对每一选择用规则 4～8 尽可能将其移向语法树的叶端；

步骤 3 把投影移自叶端，应注意：

$$\Pi_A(\sigma_F(r))\Rightarrow\Pi_A(\sigma_F(\Pi_{AB}(r))$$

其中，F 涉及 A 和 B。

步骤 4 把连串的选择（或投影）合并起来。应注意：$\sigma_{F_1\times F_2\times\cdots F_n}(r)\Rightarrow\sigma_{F_1}(\sigma_{F_2}(\cdots(\sigma_{F_n}(r))\Rightarrow\sigma_{F_1\wedge F_2}(\cdots\sigma_{F_n}(r))$，也就是将不能向叶端移动的选择合并起来一次做。

步骤 5 分组（目的是并行）。

分组的原则是：

1）每一个双目运算（$\times$，$\bowtie$，$\cup$，$-$）与其直接祖先分为一组。

2）若后代结点直至叶子全为单目运算，则将这一串单目运算并为该组。

3) 若“×”运算的后代结点不能与它结合成等值连接，则不能将其后代结点与“×”运算归为同一组。

步骤 6　每组中结点计算作为一步，各步的顺序是任意的，但需保证每一步的计算在它的后代组计算之后进行。

由此，我们可以得到优化后的计算表达式。我们用例子说明。

例 1.2.15　借书人模式为 BW，图书模式 BK 和借阅表模式 L 如下：

BW

姓名(Name)	借书证号(Card#)	单位(Dep)

BK

图书馆号 L#	书名(Title)	作者(Auther)	价格(Price)	出版社(p#)

L

借书证(Card#)	图书馆号 L#	日期(Date)

在模式下用户视图的子模式 subs 为

$$\text{subs}=\Pi_S(\sigma_F(\text{BW}\times\text{BK}\times\text{L}))$$

其中

$$F=\{\text{BW}\cdot\text{C}^{\#}=\text{L}\cdot\text{C}^{\#}\wedge\text{L}\cdot\text{L}^{\#}=\text{BK}\cdot\text{L}^{\#}\}$$

$$S=\{\text{Title},\text{Auther},\text{L}^{\#},\text{Name},\text{Dep},\text{Card}^{\#},\text{Date}\}$$

查询要求是：列出 1990 年以后借出的书名。

初始语法树为图 1.2.2。

优化过程分两步：

1) 将 $\sigma_{\text{BW}\cdot\text{C}^{\#}=\text{L}\cdot\text{C}^{\#}\wedge\text{L}\cdot\text{L}^{\#}=\text{BK}\cdot\text{L}^{\#}}\Rightarrow\begin{cases}\sigma_{\text{BW}\cdot\text{C}^{\#}=\text{L}\cdot\text{C}^{\#}}\\ \sigma_{\text{L}\cdot\text{L}^{\#}=\text{BK}\cdot\text{L}^{\#}}\end{cases}$

2) 将选择拉到叶端

其过程如图 1.2.3 所示。

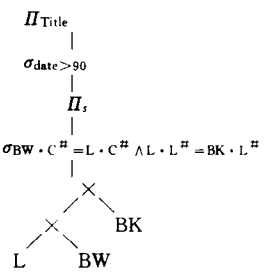
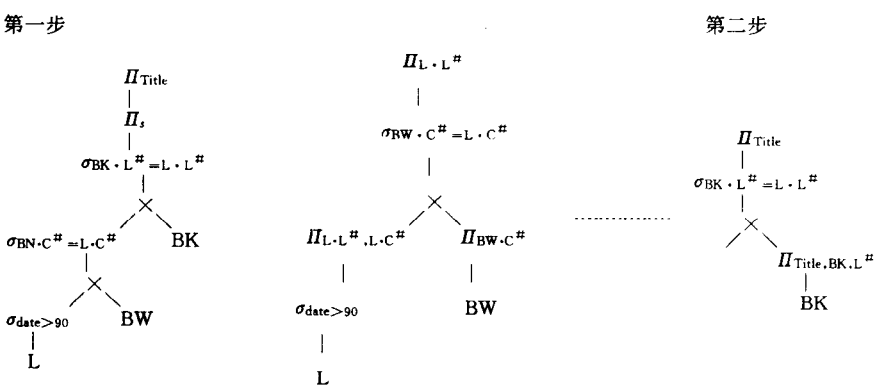


图 1.2.2　初始语法树



若为非等值的，如“>”则“×”，以下的单目运算不能并入该组，应另成一组

图 1.2.3　优化过程

1.3　数据依赖

我们知道，关系的表现形式是一张二维表格。表头由实体的特征——属性构成。我们将

这些属性的集合称为关系模式。这些属性之间并不是相互独立，互不相关的，它们共同为描述现实世界的某些实体服务。进一步，现实世界的许多客观事实限制了关系模式的所有关系实例必须满足一定的约束条件。某些条件通过对属性的取值范围的限定得以满足，例如，某大学的学生的年龄必须大于 14 岁。另一种条件的满足方法是通过属性值之间的相互关联的关系（主要体现在取值的相等或不等）得以满足。后一种满足的条件我们称之为数据依赖，即属性之间的关系是属性之间的互相关联，互相制约，互相依赖的关系。它通过一个关系中属性间的取值相等与否来体现数据之间的相互关系，是现实世界属性间关联的一种抽象，是数据之间的内在联系和语义的体现。在这里，我们将讨论三种最为重要的数据依赖：函数依赖（Functional Dependency 记为 FD），多值依赖（Multivalued Dependency，记为 MVD），和连接依赖（Join Dependency，记为 JD）。

1.3.1 函数依赖

1. 函数依赖的概念

函数依赖普遍存在于现实世界的数据之中，例如，当我们对一个学生实体进行描述时，学生实体被描述为一个关系模式，记为 Stud＝{学号（S#），姓名（Sn），性别（Se），年龄（Old）}。由于学号与学生是一一对应的关系，因而，当我们确定了学号（S#）属性的一个具体值，如“9713049”，之后，就可以惟一地确定这个学生是谁。与之相对应的一系列其它特征，如姓名、性别等，也就相应地被确定了。这与数学中的函数关系类似，给定了一个自变量 X，就得到与之对应的函数值 $f(X)$。在这里，我们称“学号（S#）函数决定了“Sn”和“Old”，或称“Sn”和“Old”函数依赖于“S#”，将此记为“S#→Sn 及 S#→Old”。

通过上述例子我们对函数依赖有了直观的认识。设 X,Y 是两个属性集，若对属性集 X 的每一个具体的值都有 Y 的惟一具体值与之对应，称 X 函数决定于 Y 或称 Y 函数依赖于 X。

上述直观地定义对属性之间的客观语义联系作出了描述，但它不能为进一步研究问题作形式化的准备，因此，我们给出以下的形式化定义。

定义 1.3.1 设 $R(U)$ 是属性集 U 上的一个关系模式，X,Y 是 U 的子集。设 r 是 R 上的一个关系实例。t_i,t_j 是 r 中任意两个元组。若 $t_i[X]=t_j[X]$，则必有 $t_i[Y]=t_j[Y]$ 成立，则称 r 满足函数依赖 $X\rightarrow Y$。

例 1.3.1 A,B,C 是 U 上的属性，r 为一个关系实例，如表 1.3.1 所示。

从例 1.3.1 看出，实例 r 满足函数依赖 $A\rightarrow B$，但不满足 $B\rightarrow C$。

定义 1.3.1 是站在检验实例对于客观语义的满足性的立场来刻划函数依赖的。同时，我们注意到蕴涵条件为假则命题恒真。即若不存在 $t_i[A]=t_j[A]$，无论 $t_i[B]$ 是否与 $t_j[B]$ 相等，命题恒为真。

表 1.3.1 实例 r

元组号	A	B	C
t_1	a	b	c
t_2	a	b	d
t_3	c	e	d
t_4	c	e	e

一般地，可将一个关系模式定义为一个二元组 $R=(U,F)$。F 为给定的一组函数依赖的集合，U 为属性集。下面给出一个算法，用于检验 U 上的某个关系实例 r 是否满足整个函数依赖集 F。

算法 1.3.1 检验 r 对 F 的可满足性。

输入：U 上的关系实例 r 与 U 上的函数依赖集 F；

输出：r 满足 F，输出 True，否则 False。

```
BEGIN
FOR 对每个 F 中的 X→Y  DO
  IF  t_i[X]=t_j[Y]且 t_i[Y]≠t_j[Y]  THEN
      BEGIN
          WRITE('False');
          GO  TO  L1;
      END;
WRITE('True');
L1;
END
```

由算法可知，函数依赖是属性间客观语义联系的描述，是否满足这种语义上的约束是检验数据正确性的一个标准。函数依赖的形式化定义为我们检验一个实例是否满足客观的语义联系提供了手段和方法。

表 1.3.2　实例 r

元组号	X	Y	Z	W
t_1	a	d	b	a
t_2	a	d	b	b
t_3	b	c	b	c
t_4	b	c	b	d
t_5	c	e	a	e

例 1.3.2　给定 $U=\{X,Y,Z,W\}$，函数依赖集 $F=\{X\rightarrow Y,Y\rightarrow Z\}$，给定下列的关系实例 r 如表 1.3.2 所示。

显然，r 满足函数依赖集 F。

上例中，可观察到 r 还满足 $X\rightarrow Z$，但在函数依赖集 F 中没有 $X\rightarrow Z$。那么，我们就需要考虑 r 除满足 F 之外，是否还会满足一些其它的函数依赖？

2. Armstrong 公理

定义 1.3.2　函数依赖的逻辑蕴涵。

设 $R\langle U,F\rangle$ 是受函数依赖集 F 约束的 U 上的关系模式，r 为 U 上的满足 F 的任意关系实例，若函数依赖 $X\rightarrow Y$ 在 r 上也成立，则称 F 逻辑蕴涵 $X\rightarrow Y$，记为 $F\models X\rightarrow Y$。

在上面的例子中，r 实例除满足 F 外，还满足 F 所不包括的 $X\rightarrow Z$，这就产生一个问题，如何从 $F=\{X\rightarrow Y,Y\rightarrow Z\}$ 推导得出 $X\rightarrow Z$，这需要一套规则从给定的函数依赖集推出其蕴涵的函数依赖。Armstrong 于 1974 年给出了一组这样的推理规则。

Armstrong 公理

A_1：若 $Y\subseteq X$，则 $X\rightarrow Y$　　（自反律，reflexivity）

A_2：若 $X\rightarrow Y$，则 $XZ\rightarrow YZ$　　（增广律，augmentation）

A_3：若 $X\rightarrow Y$ 且 $Y\rightarrow Z$，则 $X\rightarrow Z$　　（传递律，transitivity）

在这里，由自反律得到的函数依赖称为平凡函数依赖，这意味着无论 r 实例如何，这样的函数依赖都是成立的。自反律并不依靠函数依赖集 F，任意的关系实例都满足自反律。

定理 1.3.1　Armstong 公理是正确的。

证明

(1) A_1 是正确的

设 $Y\subseteq X$，r 为 $R\langle U,F\rangle$ 上的任意关系实例，t_i,t_j 为 r 中任意两个元组。若 $t_i[X]=t_j[X]$，

由 $Y\subseteq X$，可知 $t_i[Y]=t_j[Y]$。所以 r 也满足 $X\rightarrow Y$，故 A_1 成立。

(2) A_2 是正确的

设 r 为 $R\langle U,F\rangle$ 上的任意关系实例，t_i,t_j 为 r 中任意两个元组。假定 $t_i[XZ]=t_j[XZ]$。由 A_1 可知：$t_i[X]=t_j[X]$ 且 $t_i[Z]=t_j[Z]$。因为 $X\rightarrow Y$，我们得到 $t_i[Y]=t_j[Y]$。$t_i[Z]=t_j[Z]$ 及 $t_i[Y]=t_j[Y]$ 成立，即 $t_i[YZ]=t_j[YZ]$ 成立。r 满足 $XZ\rightarrow YZ$。

(3) A_3 是正确的

设 $F\models X\rightarrow Y,F\models Y\rightarrow Z$。

若 r 为 $R\langle U,F\rangle$ 上的任意关系实例，t_i,t_j 为 r 中任意两个元组，若 $t_i[X]=t_j[X]$，则由 $X\rightarrow Y$ 可知 $t_i[Y]=t_j[Y]$，再由 $Y\rightarrow Z$ 可知 $t_i[Z]=t_j[Z]$ 也成立。由 $t_i[X]=t_j[X]$ 有 $t_i[Z]=t_j[Z]$，即 $X\rightarrow Y,Y\rightarrow Z$ 成立，则 $X\rightarrow Z$ 也成立。

综上所述，Armstrong 公理正确性得证。 □

3. Armstrong 公理的完备性

这里仍有一个问题，Armstrong 三条公理虽已证明为正确的，但它是否具有完备性？也就是说，F 逻辑蕴涵的所有依赖是否都能由 F 经过 $A_1\sim A_3$ 推导而得呢？为了对完备性问题进行进一步讨论，我们先给出以下定义。

定义 1.3.3 设 F 为 U 上的函数依赖集，由 F 经过 Armstrong 公理推导出的全体函数依赖集称为 F 的闭包，记为 F^+。

要证明 Armstrong 的完备性，只需证明 F^+ 等于 F 所逻辑蕴涵的所有依赖。

一种直观简单的思路是将 F^+ 中的所有依赖全部列出，一一考察它们是否被 F 所逻辑蕴涵，即看它们是否被 F 的任意实例 r 所满足。

然而，列出 F^+ 是件费时间的事，在某些情况下，既使函数依赖集 F 很小(所包含的函数依赖少)，F^+ 也可能很庞大，下面我们用例子说明。

例 1.3.3 设 $F=\{A\rightarrow B_1,A\rightarrow B_2,\cdots,A\rightarrow B_n\}$。

在此例中，F^+ 包含了所有形如 $A\rightarrow Y$ 的依赖，Y 是 $\{A,B_1,B_2,\cdots,B_n\}$ 的任一子集，共有 2^{n+1} 个这样的函数依赖，随着 n 的增大，F^+ 中的依赖呈指数级上升，此问题是 NP 困难性的。要枚举 F^+ 是困难的，如何解决呢？先给出以下定义：

定义 1.3.4 设 F 是 $U=\{A_1,A_2,\cdots,A_n\}$ 上的函数依赖集，$X\subseteq U$，由 F 经 $A_1\sim A_3$ 推导出的形如 $X\rightarrow A_i$ 的依赖的右部属性集称为 X 关于 F 的属性闭包，记为 X_F^+。

为使概念清晰，需注意以下两点：

1) F^+ 的元素是函数依赖，形如 $X\rightarrow Y$，而 X_F^+ 中的元素是一些属性。

2) X 一定在 X_F^+ 中，这是由自反律 $X\rightarrow X$ 而得到的。

例 1.3.4 设 $U=\{A_1,A_2,A_3\},F=\{A_1\rightarrow A_2,A_2\rightarrow A_3\}$。

若 $X=A_1$，则 $X_F^+=\{A_1,A_2,A_3\}$；

若 $X=A_2$，则 $X_F^+=\{A_2,A_3\}$；

若 $X=A_3$，则 $X_F^+=\{A_3\}$。

计算函数依赖 F 的闭包 F^+ 是困难的，而计算属性集的闭包是一种容易的事情。

下面给出具体算法求 X_F^+。

算法 1.3.2 求属性闭包。

输入：属性集 X，函数依赖集 F，

输出：X_F^+

步骤 1　$i=0$；$X^{(i)}=X$；

步骤 2　求 B；

$$B=\{A\mid(\exists\ V)(\exists\ W)\mid(V\to W\in F\wedge V\subseteq X^{(i)}\wedge A\in W\}$$

步骤 3　$X^{(i+1)}=X^{(i)}\cup B$；

步骤 4　查 $X^{i+1}=X^{(i)}$吗？

步骤 5　若不等，用 $i+1$ 替换 i，返回步骤 2；

步骤 6　若相等，$X^{(i)}=X_F^+$，算法中止。

利用此算法，我们来看一个求属性闭包的例子。

例 1.3.5　设 $F=\{AB\to C,C\to A,BC\to D,ACD\to B,D\to EG,BE\to C,CG\to BD,CE\to AG\}$

记 $X=BD$。

求$(BD)_F^+$？

解　1）$X^{(0)}=\{B,D\}$；

2）因为 $BD\supseteq D$，由 $D\to EG$ 可得 $X^{(1)}=X^{(0)}\cup EG=BDEG$，且 $X^{(1)}\neq X^{(0)}$；

3）因为 $BE\subseteq BDEG$，$BE\subseteq X^{(1)}$，由 $BE\to C$ 得 $X^{(2)}=BDEGC$，且 $X^{(2)}\neq X^{(1)}$；

4）因为 $C\subseteq X^{(2)}$，由 $C\to A$ 有 $X^{(3)}=ABCDEG$；由于 $X^{(3)}=U$，再也不可能增大，可知 $X^{(3)}=X^{(4)}$，以后不可能再增加新元素；

5）算法中止。

故$(BD)_F^+=X^{(3)}=ABCDEG$。

证明算法的正确性之前先给出引理。

引理 1.3.1　若 $X_1\subseteq X_2$，则对于一切 j 都有 $X_1^{(j)}\subseteq X_2^{(j)}$。

引理证明采用归纳法。

归纳基础：$j=0$，$X_1^{(0)}\subseteq X_2^{(0)}$，即 $X_1\subseteq X_2$，命题成立。

归纳假设：设 $j-1$ 时命题为真。即有 $X_1^{(j-1)}\subseteq X_2^{(j-1)}$。

归纳推广：由算法可知，对一切 $W\to Z\in F$，$W\in X_1^{(j-1)}$ 即有 $X_1^{(j)}=X_1^{(j-1)}\cup Z$。同样，对一切 $W'\to Y\in F$，$W'\subseteq X_2^{(j-1)}$，因为 $X_2^{(j-1)}\supseteq X_1^{(j-1)}$（归纳假设），有 $X_2^{(j)}=X_2^{(j-1)}\cup Y$。由此可知 W 能推导出的，W' 也一定能推导出，所以我们有 $Z\subseteq Y$，也即 $X_2^{(j)}\supseteq X_1^{(j)}$。　□

该引理对证明 X_F^+ 中任意属性 Y 必在某个 $X^{(k)}$ 之中是很有用的。

定理 1.3.2　算法 1.3.2 是正确的。

分析　要证明算法是正确的，我们需要证明以下两个命题。

命题 1　$X_F^+\supseteq X^{(k)}$

命题 2　$X_F^+\subseteq X^{(k)}$

证明

(1) 采用数学归纳法证明命题 1

归纳基础：$j=0$，$X^{(0)}=X$，$X\subseteq X_F^+$，结论为真。

归纳假设：设 $j=k-1$ 时结论为真，即有 $X^{k-1}\subseteq X_F^+$。

归纳推广：由算法可知，$X^{(k)}=X^{(k-1)}\cup Z$，其中 $W\to Z\in F$ 且 $W\subseteq X^{k-1}$，由归纳假定 X^{k-1}