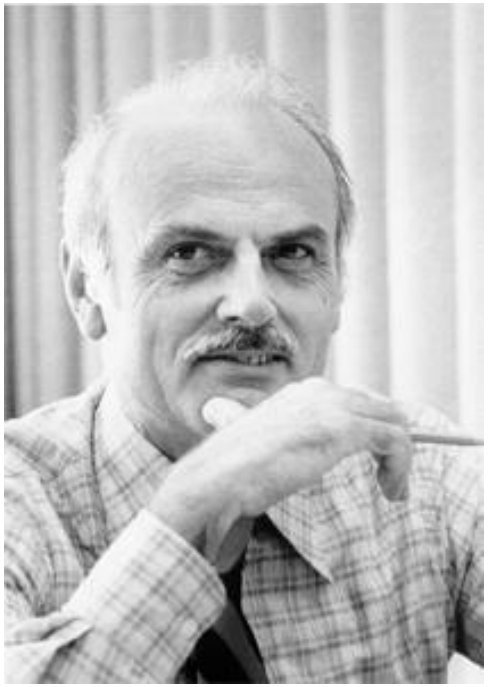
第二章　关系数据库

E.F.Codd于70年代初提出关系数据模型与关系数据理论，因此获得1981年的ACM图灵奖。



Edgar Frank Codd，1923－2003

Information Retrieval P. BAXENDALE, Editor

A Relational Model of Data for Large Shared Data Banks

E. F. CODD
IBM Research Laboratory, San Jose, California

Future users of large data banks must be protected from having to know how the data is organized in the machine (the internal representation). A prompting service which supplies such information is not a satisfactory solution. Activities of users at terminals and most application programs should remain unaffected when the internal representation of data is changed and even when some aspects of the external representation are changed. Changes in data representation will often be needed as a result of changes in query, update, and report traffic and natural growth in the types of stored information.

Existing noninferential, formatted data systems provide users with tree-structured files or slightly more general network models of the data. In Section 1, inadequacies of these models are discussed. A model based on n-ary relations, a normal form for data base relations, and the concept of a universal data sublanguage are introduced. In Section 2, certain operations on relations (other than logical inference) are discussed and applied to the problems of redundancy and consistency in the user's model.

KEY WORDS AND PHRASES: data bank, data base, data structure, data organization, hierarchies of data, networks of data, relations, derivability, redundancy, consistency, composition, join, retrieval language, predicate calculus, security, data integrity
CR CATEGORIES: 3.70, 3.73, 3.75, 4.20, 4.22, 4.29

1. Relational Model and Normal Form

1.1. INTRODUCTION

This paper is concerned with the application of elementary relation theory to systems which provide shared access to large banks of formatted data. Except for a paper by Childs [1], the principal application of relations to data systems has been to deductive question-answering systems. Levein and Maron [2] provide numerous references to work in this area.

In contrast, the problems treated here are those of data *independence*—the independence of application programs and terminal activities from growth in data types and changes in data representation—and certain kinds of data *inconsistency* which are expected to become troublesome even in nondeductive systems.

The relational view (or model) of data described in Section 1 appears to be superior in several respects to the graph or network model [3, 4] presently in vogue for noninferential systems. It provides a means of describing data with its natural structure only—that is, without superimposing any additional structure for machine representation purposes. Accordingly, it provides a basis for a high level data language which will yield maximal independence between programs on the one hand and machine representation and organization of data on the other.

A further advantage of the relational view is that it forms a sound basis for treating derivability, redundancy, and consistency of relations—these are discussed in Section 2. The network model, on the other hand, has spawned a number of confusions, not the least of which is mistaking the derivation of connections for the derivation of relations (see remarks in Section 2 on the "connection trap").

Finally, the relational view permits a clearer evaluation of the scope and logical limitations of present formatted data systems, and also the relative merits (from a logical standpoint) of competing representations of data within a single system. Examples of this clearer perspective are cited in various parts of this paper. Implementations of systems to support the relational model are not discussed.

1.2. DATA DEPENDENCIES IN PRESENT SYSTEMS

The provision of data description tables in recently developed information systems represents a major advance toward the goal of data independence [5, 6, 7]. Such tables facilitate changing certain characteristics of the data representation stored in a data bank. However, the variety of data representation characteristics which can be changed *without logically impairing some application programs* is still quite limited. Further, the model of data with which users interact is still cluttered with representational properties, particularly in regard to the representation of collections of data (as opposed to individual items). Three of the principal kinds of data dependencies which still need to be removed are: ordering dependence, indexing dependence, and access path dependence. In some systems these dependencies are not clearly separable from one another.

1.2.1. *Ordering Dependence.* Elements of data in a data bank may be stored in a variety of ways, some involving no concern for ordering, some permitting each element to participate in one ordering only, others permitting each element to participate in several orderings. Let us consider those existing systems which either require or permit data elements to be stored in at least one total ordering which is closely associated with the hardware-determined ordering of addresses. For example, the records of a file concerning parts might be stored in ascending order by part serial number. Such systems normally permit application programs to assume that the order of presentation of records from such a file is identical to (or is a subordering of) the

Volume 13 / Number 6 / June, 1970 Communications of the ACM 377

- 关系理论是建立在集合代数理论基础上的，有着坚实的数学基础。
- 早期代表系统
  - System R：由IBM研制。
  - INGRES：由加州Berkeley分校研制。
- 目前主流的商业数据库系统
  - Oracle，MySQL， DB2 ， SQL- Server等

主要内容

- 关系模型的基本概念
- 关系代数
- 元组关系演算与域关系演算
- 三类关系运算的安全约束及等价性
- 关系数据语言概述

关系的数学定义

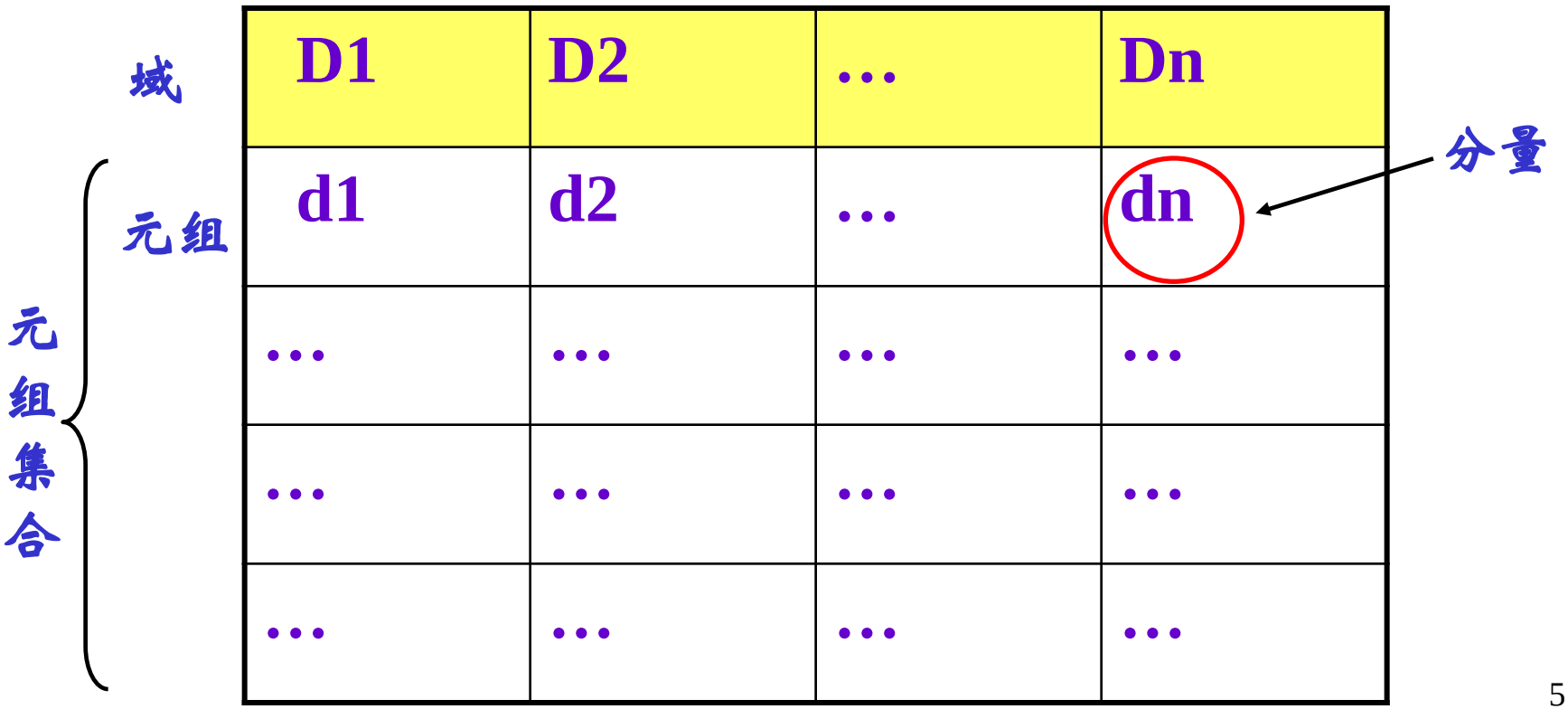
- 域（Domain）
  - 定义：一组具有相同数据类型值的集合。例如，整数，实数，{0，1}等。
- 元组和分量
  - 定义：给定一组域$D_1, D_2, \ldots, D_n$，这些域中可以有相同的。$D_1, D_2, \ldots, D_n$的笛卡尔积（Cartesian Product）为：
    $D_1 \times D_2 \times \ldots \times D_n = \{(d_1, d_2, \ldots, d_n) \mid d_i \in D_i, i=1,\ldots,n\}$。
    笛卡尔积的每个元素$(d_1, d_2, \ldots, d_n)$称作一个n元组（n-tuple）或简称元组。元组的每一个值d_i叫做一个分量（component）。

若D_i（i=1,…,n）为有限集，其基数（Cardinal number）为m_i，则笛卡尔积的基数为 $\prod_{i=1}^{n} m_i$

关系的数学定义

- 笛卡儿积可以表达为二维表：

$$D1 \times D2 \times \ldots \times Dn = \{(d1, d2, \ldots, dn) \mid di \in Di, i=1,\ldots,n\}$$



笛卡儿积示例

例：有三个域　　D_1=MAN={M_1，M_2，M_3}

D_2=WOMAN={W_1，W_2}

D_3=CHILD={C_1，C_2，C_3}

则 $D_1 \times D_2 \times D_3$ =

{（M_1，W_1，C_1），（M_1，W_1，C_2），

（M_1，W_1，C_3），……}

二维表的表示：

D_1	D_2	D_3
M_1	W_1	C_1
M_1	W_1	C_2
M_1	W_1	C_3
…	…	…

共18个

关系的定义

- 关系的定义：笛卡尔积$D_1 \times D_2 \times \ldots \times D_n$的子集叫做在域$D_1, D_2, \ldots, D_n$上的关系，用$R(D_1, D_2, \ldots, D_n)$表示。R是关系的名字，n是关系的度或目(Degree)。

  当n=1时称为单元关系，

  当n=2时称为二元关系，依此类推。

- 关系可以表示为二维表：
  - 表的框架由D_i（i=1,2,…,n）构成
  - 表的每一行对应一个元组
  - 表的每一列对应一个域
  - 每个列附加一个名称，则每个列称为一个属性(Attribute)。

    属性的名字是唯一的。属性A_i的取值范围D_i（i=1,2,…,n）称为值域。 n元关系必有n个属性。

A_1	A_2	…	A_n
a1	a2	…	an
…	…	…	…
…	…	…	…

关系示例

以MAN、WOMAN、CHILD三个域的笛卡儿积的子集构造FAMILY关系：

FAMILY（MAN，WOMAN，CHILD）

其中MAN—丈夫，WOMAN—妻子，CHILD—这对夫妻的子女。假设M1、W1、C1，M2、W2、C2，C3是两个家庭，则FAMILY关系为：

属性

元组

MAN	WOMAN	CHILD
M_1	W_1	C_1
M_2	W_2	C_2
M_2	W_2	C_3

关系的性质

- 列是同质的（Homogeneous）即每一列中的分量来自同一域，是同一类型的数据；
- 不同的列可出自同一域，每列必须有不同的属性名；
- 列的顺序无所谓，即列次序可以互换；
- 任意两个元组不能完全相同；
- 行的顺序无所谓，即行次序可以互换；
- 每一分量必须是不可再分的数据。满足这一条件的关系称作满足第一范式（1NF）的。

FAMILY

MAN	WOMAN	CHILD
M_1	W_1	C_1
M_2	W_2	C_2
M_2	W_2	C_3

关系数据模型

- 关系模型的数据结构
- 关系模型的语义约束
- 关系模型的数据操作概述

关系模型的数据结构（1）

- 关系模型的数据结构—关系
  - 实体及实体之间的联系均用单一的数据结构—“关系”来表示。
- 几个基本概念
  - 关系、域、n目关系、元组、属性。
  - 码（Key,键）
    - 侯选码（Candidate key）：关系中的某一属性组，若它的值唯一地标识了一个元组，并具有最小性，则称该属性组为侯选码。
    - 主码(Primary key,首码，码）：若一个关系有多个侯选码，则选定其中一个为主码。
  - 主属性与非主属性
    - 码中的诸属性称为主属性，不包含在任何候选码中的属性称为非主属性。

关系模型的数据结构（2）

U为组成该关系的属性名集合

D为属性集U所来自的域

dom为属性向域的映象集合

F为属性间的数据依赖关系集合

I为完整性约束集合

关系模式

- 关系的描述称作关系模式，它可以形式化地表示为：

  R（U，D，dom，F，I）

- 关系模式通常可以简记作$R(A_1, A_2, \ldots, A_n)$，R为关系名，A_1，A_2，…，A_n为属性名，D和dom直接说明为属性的类型、长度。

- 关系是关系模式在某一时刻的状态或内容。关系模式是相对稳定的。而关系是动态的，是随时间不断变化的。

—关系数据库（关系数据库的型和值的概念）

- 关系模式的集合构成关系数据库模式—关系数据库的型；
- 关系的集合则构成具体的关系数据库—关系数据库的值。

关系模型的语义约束

- 实体完整性
- 参照完整性

} 关系模型必须支持的约束条件

- 用户定义完整性
  - 用户针对具体的应用环境定义的完整性约束条件。

关系模型的语义约束

- 实体完整性（Entity Integrity）
  - 要有属性或属性组合作为主码，主码值不可为空或部分为空。或定义为若属性A是关系R的主属性，则属性A不能取空值。
  - 空值的含义是：不知道或不存在的值。

学生

学号	姓名	年龄	系别
s1	A	18	CS
s2	B	18	CS
s3	C	18	MA

关系模型的语义约束

学生

学号	姓名	年龄	系别
s1	A	18	CS
s2	B	18	CS
s3	C	18	MA

课程

课号	课名	先行课号
c1	aaa	
c2	bbb	c1
c3	ccc	

学生选课

学号	课号	成绩
s1	c1	80
s1	c2	90
s2	c1	95

- 参照完整性（Referential integrity）
  - 外部码
    - 设F是基本关系R的一个或一组属性，但不是R的码。如果F与基本关系S的主码Ks相对应，则称F是关系R的外部码（Foreign Key），并称R为参照关系（Referencing Relation），S为被参照关系（Referenced Relation）或目标关系（Target Relation）。R和S不一定是不同的关系。
    - 目标关系S的主码Ks和参照关系的外部码F必须定义在一个域上。
  - 参照完整性
    - 如果关系R的外部码F_k与关系S的主码P_k相对应，则R中的每一个元组的F_k值或者等于S中某个元组的P_k值，或者为空值。

关系模型的语义约束

- 参照完整性示例

职工关系EMP（ENO，ENAME，DNO）和部门关系DEPT（DNO，DNAME）是两个基本关系。

DNO是EMP的外码。

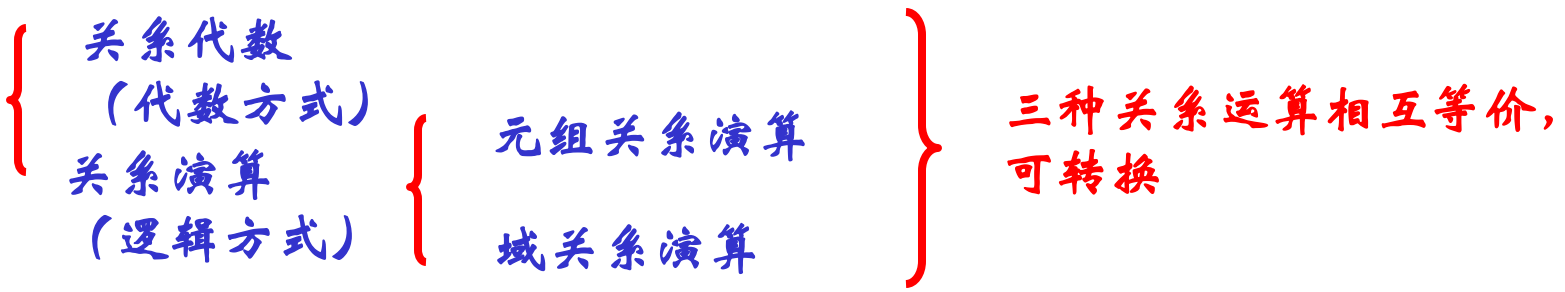
EMP中每个元组在DNO上的取值允许有两种：

（1）取空值；

（2）非空，则DNO的值必须是DEPT中某个元组的DNO值。

关系模型的数据操作

- 关系数据操作方式的特点是集合操作，“一次一集合”方式。操作的对象与结果都是集合。
- 关系数据操作的基础是“关系运算”。关系运算方式有两种：代数方式，逻辑方式。



关系代数简介

- 关系代数是三种关系运算中的基础方法，有9种：
  - 常规集合运算：并、差、交、广义笛卡儿积（乘）；
  - 特有的关系运算：选择、投影、连接、自然连接、求商。

关系演算简介

- 元组演算表达式的形式：

  $\{t \mid \Phi(t)\}$，

  其中 t 为元组变量，$\Phi(t)$是元组关系演算公式，由原子公式和运算符组成；

- 域关系演算表达式的形式：

  $\{(x_1,x_2,\ldots,x_k) \mid \Phi(x_1,x_2,\ldots,x_k)\}$

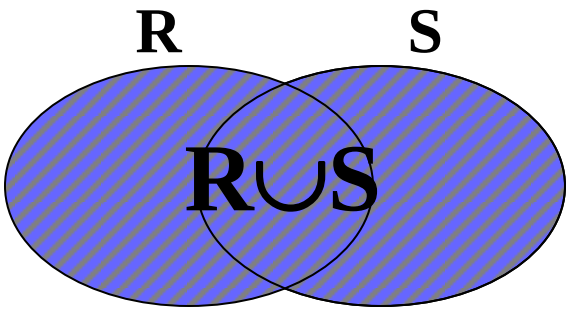
其中x_i代表域变量，Φ为域关系演算公式，由原子公式和运算符组成。

关系代数—传统集合运算

- 传统的集合运算是二目运算。除笛卡儿积外，要求参加运算的两个关系必须是**同类关系**，即两个关系具有相同的度“n”，且相对应的属性值必须取自同一个域。
- 设关系R和S是同类关系，可定义关系的集合运算如下：

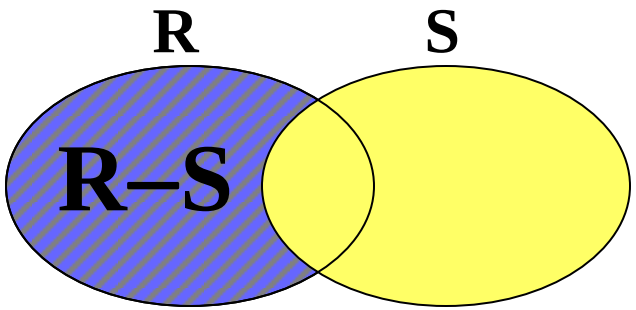
关系代数—传统集合运算

- 并（Union)
  - 关系R和关系S的并记为R∪S，结果仍为n元关系，由属于R或属于S的元组构成。
  - R∪S表达为 R∪S ={ t |t ∈R ∨ t∈S }



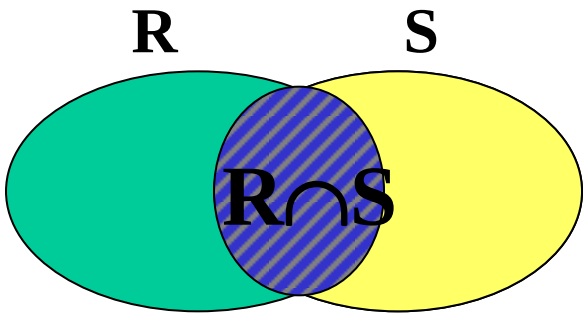
关系代数—传统集合运算

- 差（Difference)
  - 关系R和关系S的差记为R-S。结果仍为n元关系，由属于R而不属于S的元组构成。
  - R-S表达为$R-S=\{t \mid t \in R \wedge t \notin S\}$



关系代数—传统集合运算

- 交（Intersection)
  - 关系R和关系S的交记为R∩S。结果仍为n元关系，由既属于R又属于S的元组构成。
  - R∩S表达为R∩S ={ t | t∈R ∧ t∈S }



关系代数—传统集合运算

- 广义笛卡儿积（Extended cartesian product）
  - 两个关系R，S，其度分别为n，m，则它们的笛卡尔积是所有这样的元组集合：元组的前n个分量是R中的一个元组，后m个分量是S中的一个元组。 R×S的度为（n+m）。

    若R有k_1个元组，S有k_2个元组，则R×S有

    $k_1 \times k_2$个元组。
  - R×S 表达为$R \times S=\{ t \mid t=< r , s > \wedge r \in R \wedge s \in S \}$

关系代数——传统集合运算

- 示例

R

A	B	C
a1	b1	c1
a1	b2	c2
a2	b2	c1

S

A	B	C
a1	b2	c2
a1	b3	c2
a2	b2	c1

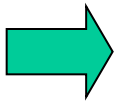
求：R∪S　　R-S　　R∩S　　R×S

关系代数—专门的关系运算

- 选取或限制（Selection or Restriction）
  - 定义：在关系R中选择满足给定条件的元组。记作：
    $\sigma_F(R)=\{t \mid t \in R, F(t)=\text{‘真’}\}$
    - F是选择的条件，取逻辑值“真”或“假”。
    - F由运算对象（属性名、常数、简单函数），运算符，包括算术比较符（$>, \geq, <, \leq, =, \neq$)和逻辑运算符（$\wedge$，$\vee$，$\neg$）连接起来的表达式组成。

SC

S#	C#	G
95001	C1	92
95001	C2	85
95001	C3	88
95002	C2	90
95002	C3	80



SC1

S#	C#	G
95001	C1	92
95001	C2	85
95001	C3	88

示例

学生表 S

S#	SN	SA	SD
95001	李勇	20	CS
95002	刘晨	19	IS
95003	王敏	18	MA
95004	张立	19	IS

学生选课表SC

S#	C#	G
95001	C1	92
95001	C2	85
95001	C3	88
95002	C2	90
95002	C3	80

课程表C

C#	CN	PC#
C1	数据库	C5
C2	数学	
C3	信息系统	C1
C4	操作系统	C6
C5	数据结构	C7
C6	数据处理	
C7	Pascal语言	C6

S(S#,SN,SA,SD);
C(C#,CN,PC#);
SC(S#,C#,G)

- 例1：检索计算机科学系（CS）学生的信息

  $\sigma_{SD='CS'}(S)$

- 例2：检索C1课程的选修情况

  $\sigma_{C\#='C1'}(SC)$

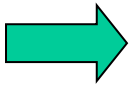
关系代数—专门的关系运算

- 投影（Projection)
  - 定义：从关系R(U)中取若干属性列并删去重复行，组成新的关系。记作：

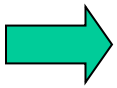
$$\Pi_A(R) = \{ t[A] \mid t \in R , A \subseteq U \}$$

SC

S#	C#	G
95001	C1	92
95001	C2	85
95001	C3	88
95002	C2	90
95002	C3	80



S#
95001
95001
95001
95002
95002



S’

S#
95001
95002

S(S#,SN,SA,SD);
C(C#,CN,PC#);
SC(S#,C#,G)

- 例1：查询学生的姓名和所在系

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

关系代数—专门的关系运算

- 连接(Join)
  - 定义：关系R和S在属性X和Y上的连接（X、Y是连接属性，即X、Y包含同等数量的属性，且相应的属性均具有共同的域），是从两个关系的广义笛卡儿积R×S中选取给定属性(X和Y)间满足θ比较条件的元组。记作：

$$R \underset{X\theta Y}{\bowtie} S = \{t \mid t = <r, s> \wedge r \in R \wedge s \in S \wedge r[X]\ \theta\ s[Y]\}$$

其中：$\bowtie$是连接运算符；

θ是算术比较运算符，

该连接也称为θ连接

当θ为“=”时，称为等值连接；

当θ为“〈”时，称为小于连接；

当θ为“〉”时，称为大于连接；

S

S#	SN	SA	SD
95001	李勇	20	CS
95002	刘晨	19	IS
95003	王敏	18	MA
95004	张立	19	IS

SC

S#	C#	G
95001	C1	92
95001	C2	85
95001	C3	88
95002	C2	90
95002	C3	80

SCS

S#	SN	SD	C#	G
95001	李勇	CS	C1	92
95001	李勇	CS	C2	85
95001	李勇	CS	C3	88
95002	刘晨	IS	C2	90
95002	刘晨	IS	C3	80

连接示例

学生表 S

S#	SN	SA	SD
95001	李勇	20	CS
95002	刘晨	19	IS
95003	王敏	18	MA
95004	张立	19	IS

$S \bowtie SC$

$S.S\# = SC.S\#$

S.S#	SN	SA	SD	SC.S#	C#	G
95001	李勇	20	CS	95001	C1	92
95001	李勇	20	CS	95001	C2	85
95001	李勇	20	CS	95001	C3	88
95002	刘晨	19	IS	95002	C2	90
95002	刘晨	19	IS	95002	C3	80

学生选课表SC

S#	C#	G
95001	C1	92
95001	C2	85
95001	C3	88
95002	C2	90
95002	C3	80

关系代数—专门的关系运算

S(S#,SN,SA,SD);
C(C#,CN,PC#);
SC(S#,C#,G)

- 自然连接（Natural Join）
  - 定义：关系R与关系S的自然连接，是从两个关系的广义笛卡儿积R×S中选取在相同属性列上取值相等的元组，并去掉重复的属性列。记作：R⋈S, 或R*S

$$R \bowtie S = \{(Z,X,W) \mid (Z,X) \in R \wedge (W,X) \in S \wedge r[X]=s[X]\}$$

  - 连接与自然连接：在等值（θ取“=”）连接情况下，连接属性X和Y是相同属性时，R与S的连接称为自然连接。自然连接的结果要在上述R与S的等值连接结果基础上再进行投影运算，去掉重复的属性列。

连接与自然连接示例

学生表 S

S#	SN	SA	SD
95001	李勇	20	CS
95002	刘晨	19	IS
95003	王敏	18	MA
95004	张立	19	IS

$S \bowtie SC$
$S.S\# = SC.S\#$

S.S#	SN	SA	SD	SC.S#	C#	G
95001	李勇	20	CS	95001	C1	92
95001	李勇	20	CS	95001	C2	85
95001	李勇	20	CS	95001	C3	88
95002	刘晨	19	IS	95002	C2	90
95002	刘晨	19	IS	95002	C3	80

学生选课表SC

S#	C#	G
95001	C1	92
95001	C2	85
95001	C3	88
95002	C2	90
95002	C3	80

$S \bowtie SC$

S.S#	SN	SA	SD	C#	G
95001	李勇	20	CS	C1	92
95001	李勇	20	CS	C2	85
95001	李勇	20	CS	C3	88
95002	刘晨	19	IS	C2	90
95002	刘晨	19	IS	C3	80

示例

- 连接与自然连接示例

R

A	B	C
a1	b1	5
a2	b3	8

S

B	E
b1	3
b3	7
b3	2

则 $R \underset{C<E}{\bowtie} S$

A	R.B	C	S.B	E
a1	b1	5	b3	7

$R \bowtie S$

A	B	C	E
a1	b1	5	3
a2	b3	8	7
a2	b3	8	2

关系代数—专门的关系运算

- 除法
  - 定义：设关系R（X，Y）与关系S（Z），其中Y和Z具有相同的属性数，且对应属性出自相同域。关系R除以关系S所得的商关系是关系R在属性X上投影的一个子集，该子集和 S（Z）的笛卡儿积必须包含在R（X，Y）中。

记为：

$R \div S=\{t|t\in \Pi_x(R) \wedge s \in S \wedge <t, s> \in R\}$

R1

S#	C#
95001	C1
95001	C2
95001	C3
95002	C2
95002	C3

÷

R2

C#
C1
C2
C3

R3

S#
95001

示例

- 例：

R

A	B	C
b	2	d
b	3	b
c	2	d
d	4	b

S

B	C
2	d
3	b

则：R ÷ S

A
b

关系代数综合示例

S(S#,SN,SA,SD);
C(C#,CN,PC#);
SC(S#,C#,G)

- 例1：求选修了C2课程的学生的学号。

$$\prod_{S\#}(\sigma_{C\#='C2'}(SC))$$

- 例2：求选修“数据库系统原理”的学生的学号。

$$\prod_{S\#}(\sigma_{CN='数据库系统原理'}(C) \bowtie SC)$$

- 例3：求至少选修了这样一门课程的学生姓名，这门课的直接先行课为C2。

$$\prod_{SN}(\sigma_{PC\#='C2'}(C) \bowtie SC \bowtie \prod_{S\#,\,SN}(S))$$

- 例4：求选修了全部课程的学生学号和姓名。

$$\prod_{S\#,\,C\#}(SC) \div \prod_{C\#}(C) \bowtie \prod_{S\#,\,SN}(S)$$

- 思考

R1

S#	C#
95001	C1
95001	C2
95001	C3
95002	C2
95002	C3

÷

R2

C#
C1
C2
C3

R3

S#
95001

S#	C#	G
95001	C1	92
95001	C2	85
95001	C3	88
95002	C2	90
95002	C3	80

÷

C#
C1
C2
C3

= ?

关系代数小结

- 从数学角度，基本关系代数运算有5种：并、差、乘、选择、投影
- 从数据库角度，核心的关系代数运算为：选择、投影、连接（或自然连接）

选择

S#	C#	G
95001	C1	92
95001	C2	85
95001	C3	88
95002	C2	90
95002	C3	80

S#	C#	G
95001	C1	92
95001	C2	85
95001	C3	88

投影

S#	C#	G
95001	C1	92
95001	C2	85
95001	C3	88
95002	C2	90
95002	C3	80

S#
95001
95002

S

S#	SN	SA	SD
95001	李勇	20	CS
95002	刘晨	19	IS
95003	王敏	18	MA
95004	张立	19	IS

SC

S#	C#	G
95001	C1	92
95001	C2	85
95001	C3	88
95002	C2	90
95002	C3	80

SCS

S#	SN	SD	C#	G
95001	李勇	CS	C1	92
95001	李勇	CS	C2	85
95001	李勇	CS	C3	88
95002	刘晨	IS	C2	90
95002	刘晨	IS	C3	80

自然连接

元组关系演算与域关系演算

- 把谓词演算应用到关系运算中就是关系演算。
  - 以元组为变量，简称元组演算；
  - 以域为变量，简称域演算。

元组关系演算

- 元组关系演算的**基本结构是元组演算表达式**。元组关系表达式的形式定义：$\{t \mid \Phi(t)\}$，表示了所有使Φ为真的元组的集合，即表示了一个关系。

其中：

- t为元组变量。如果元组变量前有“全称”（$\forall$）或“存在”（$\exists$）量词，则称其为约束元组变量，否则称为自由元组变量。
- $\Phi(t)$是元组关系演算公式，由原子公式和运算符组成，简称公式。

元组关系演算公式的递归定义（1）

(1)原子命题函数是公式，称为原子公式。原子公式有三类：

- $R(t)$。t是关系R中的一个元组。
- $t[i]\ \theta\ u[j]$。t[i]与u[j]分别为t的第i个分量和u的第j个分量，它们之间满足比较关系θ。
- $t[i]\ \theta\ c$或$c\ \theta\ t[i]$。分量t[i]与常量c之间满足比较关系θ。

(2)如果Φ_1，Φ_2是公式，则$\Phi_1 \wedge \Phi_2$，$\Phi_1 \vee \Phi_2$，$\neg\ \Phi$也是公式。

(3)如果Φ是公式，则$\exists t(\Phi)$和$\forall t(\Phi)$也是公式。

元组关系演算公式的递归定义（2）

(4)在元组演算公式中，各种运算符的优先次序为：

- 算术比较运算符最高。
- 量词次之，且∃的优先级高于∀。
- 逻辑运算符最低，且¬优先级高于∧，∧高于∨。
- 如果有括号，则括号中的运算优先级最高。

- 按照上述4个规则对元组公式进行有限次复合，可以得到元组演算的所有公式。

示例

- 例：设关系R，S，W，求如下元组表达式的结果：

R

A1	A2	A3
a	e	8
c	f	6
d	b	4
d	f	3

S

A1	A2	A3
a	e	8
b	c	5
d	b	4
d	f	6

W

B1	B2
4	x
5	d

（1）R1={t|R(t) ∧t[3] ≥4};

(2) R2={t | (∃u)(R(t) ∧W(u) ∧t[3]<u[1])};

(3) R3 = {t|R(t) ∧S(t)};

结果：

R1

A1	A2	A3
a	e	8
c	f	6
d	b	4

R2

A1	A2	A3
d	b	4
d	f	3

元组演算与关系代数的等价性

用关系演算表达五种基本运算：

- 并：$R \cup S = \{ t \mid R(t) \vee S(t) \}$
- 差：$R - S = \{ t \mid R(t) \wedge \neg S(t) \}$
- 笛卡儿积：

$R \times S = \{ t^{(n+m)} \mid (\exists u^{(n)})(\exists v^{(m)})(R(u) \wedge S(v) \wedge t[1] = u[1] \wedge \ldots \wedge t[n] = u[n] \wedge t[n+1] = v[1] \wedge \ldots \wedge t[n+m] = v[m]) \}$

- 投影：

$\prod_{i_1, i_2, \ldots, i_k}(R) = \{ t^{(k)} \mid (\exists u)(R(u) \wedge t[1] = u[i_1] \wedge \ldots \wedge t[k] = u[i_k]) \}$

- 选取：$\sigma_F (R) = \{ t \mid R(t) \wedge F' \}$，$F'$是$F$用$t[i]$代替原运算对象$i$得到的等价公式。

示例

S(S#,SN,SA,SD);
C(C#,CN,PC#);
SC(S#,C#,G)

用关系代数与元组关系演算表达下列查询：

- 例1：查询计算机系（CS）的全体学生

  $\sigma_{SD='CS'}(S)$； $\{t \mid S(t) \wedge t[4]='CS'\}$

- 例2：查询年龄小于20岁的学生

  $\sigma_{SA<20}(S)$； $\{t \mid S(t) \wedge t[3]<20\}$

- 例3：查询学生的姓名和所在的系

  $\Pi_{SN, SD}(SD)$；

  $\{t^{(2)} \mid (\exists u)(S(u) \wedge t[1]= u[2] \wedge t[2]=u[4])\}$

域关系演算

- 域关系演算类似于元组演算。不同的是公式中的变量是对应元组各个分量的域变量。
- 域演算表达式的形式定义：

  $\{(x_1x_2\ldots x_k) \mid \Phi(x_1, x_2, \ldots, x_k)\}$，表示所有使$\Phi$为真的那些$(x_1, x_2, \ldots, x_k)$组成的元组集合，即表达了一个关系。

其中：

- x_i代表域变量，如果域变量前有“全称”（$\forall$）或“存在”（$\exists$）量词，则称其为约束域变量，否则称为自由域变量。
- Φ为域关系演算公式，由原子公式和运算符组成。

域演算公式的递归定义

- 域演算有三种形式的原子命题函数或称原子公式：
  - $R(x_1, x_2, \ldots, x_k)$。$(x_1, x_2, \ldots, x_k)$是R的一个元组，x_i是域变量或常量。
  - $x_i \theta y_j$。域变量x_i与y_j之间满足比较关系θ。
  - $x_i \theta c$或$c \theta x_i$。域变量x_i与常量c之间满足比较关系θ。
- 域演算与元组关系演算具有相同的运算符，相同的公式递归定义。

示例

- 例：设关系R，S分别如下图所示，现给出域演算公式，求结果关系。

R

A1	A2	A3
d	ce	5
d	bd	2
g	ef	7
d	cd	9

S

A1	A2	A3
c	bd	7
c	he	3
b	cf	6
d	cd	9

(1)R1={xyz|R(xyz) ∧z<8 ∧x=d}

(2)R2={xyz|(R(xyz) ∨S(xyz)) ∧x ≠c ∧y≠cd}

结果： R1

A1	A2	A3
d	ce	5
d	bd	2

R2

A1	A2	A3
d	ce	5
d	bd	2
g	ef	7
b	cf	6

关系运算的安全约束

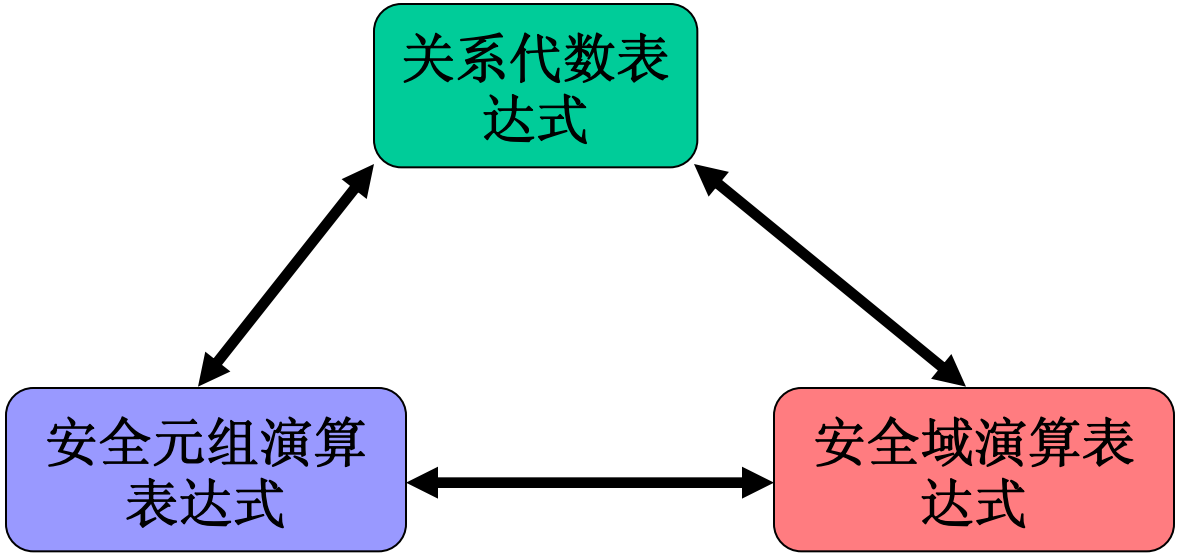
- 关系运算中把不产生无限关系和无穷验证的运算称为**安全运算**；其运算表达式称为**安全表达式**；对其所采取的限制称为**安全约束**。
- 关系代数是安全运算，关系演算则不一定是。所以对关系演算要进行安全约束。

关系运算的安全约束

- 在关系演算中，通常采用的安全约束方法是对Φ定义一个有限的符号集DOM（Φ）,使Φ的运算结果及中间结果所产生的关系及其元组的各个分量都必须属于DOM（Φ）。
- 设Φ是一个元组关系演算公式，为DOM(Φ)是由如下两类符号构成的集合：
  - Φ中的所有常量
  - Φ中出现的所有元组的所有分量值

三类关系运算的等价性

- 经过安全约束后的三类关系运算的表达能力是等价的，可以相互转换。



–每一个关系代数表达式有一个等价的安全元组演算表达式；

–每一个安全元组演算表达式有一个等价的安全域演算表达式；

–每一个安全域演算表达式有一个等价的关系代数表达式。

关系数据语言概述

- 数据库数据语言
- 关系数据语言特点
- 关系数据语言分类

数据库数据语言

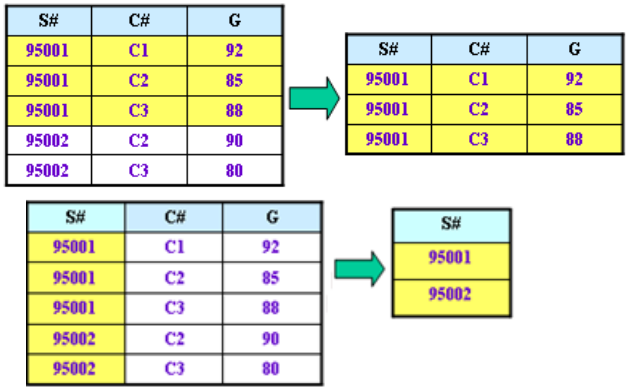
- 数据库数据语言从功能上一般分为以下几种：
  - 数据定义（描述）语言（Data definition or description language)，包括模式DDL，外模式DDL，内模式DDL；
  - 数据操纵语言（Data Manipulation Language)DML
    - 数据库有四种基本操作：检索、插入、修改、删除
    - DML有联机交互方式和宿主语言方式。
      - 联机交互方式下的DML称为自含式语言，可独立使用，适用于终端直接查询；
      - 宿主语言方式下的DML称为嵌入式语言，依附于宿主语言，是嵌入高级语言的程序中，以实现数据操作。
  - 数据控制语言(Data Control Language)DCL,完成数据库的安全性控制、完整性控制、并发控制等。

关系数据语言的特点（1）

- 一体化
  - 将数据的定义、查询、更新、控制等功能融为一体，只给用户提供一种称之为“查询语言”的语言。便于用户学习与使用。
- 非过程化
  - 用户只需提出“干什么”，而“怎样干”由DBMS解决。所以语言操作简单，易学、易用。
- 面向集合的存取方式
  - 操作对象是一个或多个关系，操作的结果也是一个新关系。
- 既可独立使用又可与主语言嵌套使用

关系数据语言的特点（2）

- 关系数据语言优越性的根源：
  - 关系模型采用了最简单、最规范化的数据结构，这使DML大大简化；
  - 关系数据语言是建立在关系运算的数学基础上，可实现关系的垂直方向和水平方向的任意分割和组装操作，使得关系语言可随机地构造出用户需要的各种各样的新关系。



S#	C#	G
95001	C1	92
95001	C2	85
95001	C3	88
95002	C2	90
95002	C3	80

S#	C#	G
95001	C1	92
95001	C2	85
95001	C3	88

S#	C#	G
95001	C1	92
95001	C2	85
95001	C3	88
95002	C2	90
95002	C3	80

S#
95001
95002

S#	SN	SA	SD
95001	李勇	20	CS
95002	刘晨	19	IS
95003	王敏	18	MA
95004	张立	19	IS

S#	C#	G
95001	C1	92
95001	C2	85
95001	C3	88
95002	C2	90
95002	C3	80

S#	SN	SD	C#	G
95001	李勇	CS	C1	92
95001	李勇	CS	C2	85
95001	李勇	CS	C3	88
95002	刘晨	IS	C2	90
95002	刘晨	IS	C3	80

关系数据语言的分类（1）

- 关系数据语言的核心是查询，所以又称为查询语言。而查询往往表示成一个关系运算表达式，因此关系运算是设计关系数据语言的基础，关系运算的分类也决定了关系语言的分类。
- 关系数据语言目前常用分类：

采用的关系运算		语言名称	
关系代数		ISBL	SQL
关系演算	元组关系演算	ALPHA；QUEL	
	域关系演算	QBE	

关系数据语言的分类（2）

- ISBL(Information System Base Language),关系代数语言的代表。英格兰底特律的IBM英国科学中心研究。应用于实验系统PRTV上。
- QUEL（Query Language），基于元组演算的语言。美国加利福尼亚大学研制。用于Ingres数据库。
- QBE(Query By Example)，基于域演算的表格显示语言。用于QBE数据库。
- SQL(Structured Query Language),介于关系代数与关系演算之间的语言，标准的关系数据语言。

本章小节

- 关系模型的基本概念
  - 关系模型的三要素
- 关系代数
  - 从数学角度，基本关系代数运算有5种：并、差、乘、选择、投影
  - 从数据库角度，核心的关系代数运算为：选择、投影、连接（或自然连接）
- 元组关系演算与域关系演算
- 关系语言的组成与特点