



第二章 关系数据库

Relational Database





本章导读

教学内容

- 关系数据模型;
- 关系的三类完整性约束;
- 基于关系运算的关系代数语言;

**请大家精准
地掌握概念**

要求掌握

- 关系数据模型的结构;
- 关系的完整性定义;
- 五种基本关系运算和四种组合关系运算;
- 用关系代数语言表达查询

教学重点及难点

- 关系代数



本章内容

关系模型概述

关系数据结构及形式化定义

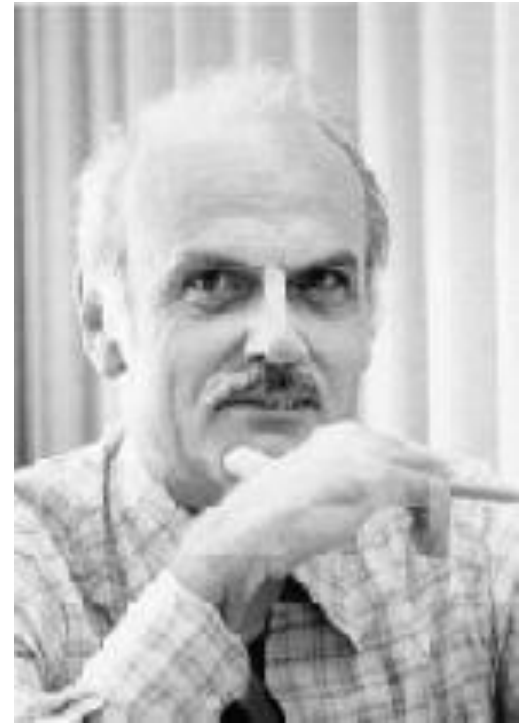
关系的完整性

关系代数



关系模型概述

关系数据模型的思想由IBM公司的E.F.Codd于1970年在他的一系列论文中提出，以后的几年里陆续出现了以关系数据模型为基础的数据库系统，称为**关系数据库系统**。



Codd published a list of 12 rules that concisely define an ideal relational database.



Codd提出的关于关系数据库的12条规则

► Rule 1: The Information Rule

All data should be presented to the user **in table form**. Last week's newsletter already discussed the basics of this rule.

► Rule 2: Guaranteed Access Rule

All data should be accessible **without ambiguity**. This can be accomplished through a **combination of the table name, primary key, and column name**.

► Rule 3: Systematic Treatment of **Null Values**

A field should be allowed to remain empty. This involves the support of a null value, which is distinct from an empty string or a number with a value of zero. Of course, this can't apply to primary keys. In addition, most database implementations support the concept of a non-null field constraint that prevents null values in a specific table column.



Codd提出的关于关系数据库的12条规则

- ▶ **Rule 4: Dynamic On-Line Catalog Based on the Relational Model** A relational database must provide **access to its structure through the same tools that are used to access the data**. This is usually accomplished by **storing the structure definition within special system tables**.
- ▶ **Rule 5: Comprehensive Data Sublanguage Rule**
The database must support at least one **clearly defined language** that includes functionality for **data definition, data manipulation, data integrity, and database transaction control**. All commercial relational databases use forms of the standard SQL (Structured Query Language) as their supported comprehensive language.
- ▶ **Rule 6: View Updating Rule**
Data can be presented to the user in different **logical combinations**, called **views**. Each view should support the same full range of data manipulation that direct access to a table has available. In practice, providing update and delete access to logical views is difficult and is not fully supported by any current database.



Codd提出的关于关系数据库的12条规则

► Rule 7: High-level Insert, Update, and Delete

Data can be retrieved from a relational database in **sets** constructed of data from multiple rows and/or multiple tables. This rule states that insert, update, and delete operations should be supported for any retrievable set **rather than just for a single row in a single table.**

► Rule 8: **Physical Data Independence**

The user is isolated from the physical method of storing and retrieving information from the database. Changes can be made to the underlying architecture (hardware, disk storage methods) without affecting how the user accesses it.

► Rule 9: **Logical Data Independence**

How a user views data should not change when the logical structure (tables structure) of the database changes. This rule is particularly difficult to satisfy. Most databases rely on strong ties between the user view of the data and the actual structure of the underlying tables.



Codd提出的关于关系数据库的12条规则

► Rule 10: **Integrity** Independence

The database language (like SQL) should support **constraints on user input** that maintain database integrity. This rule is not fully implemented by most major vendors. At a minimum, all databases do preserve two constraints through SQL. No component of a primary key can have a null value. (see rule 3) If a foreign key is defined in one table, any value in it must exist as a primary key in another table.

► Rule 11: **Distribution** Independence

A user should be totally unaware of whether or not the database is **distributed** (whether parts of the database exist in multiple locations). A variety of reasons make this rule difficult to implement; I will spend time addressing these reasons when we discuss distributed databases.

► Rule 12: Non-subversion Rule—**modify structure only through SQL**

There should be no way to modify the database structure other than through the multiple row database language (like SQL). Most databases today support administrative tools that allow some direct manipulation of the data structure. Over the life of this newsletter, I will be expanding on the concepts covered by each of Codd's rules. I will use the relational query language of choice, SQL, to illustrate these concepts and explain relational database structure in detail.



Codd提出的关于关系数据库的12条规则

- ▶ 1. 信息规则：数据以表格的形式展示给用户。
- ▶ 2. 确保访问规则：数据通过引用表的名字、主键和属性名来访问。
- ▶ 3. 空值处理：非主属性应该可以为空。
- ▶ 4. 基于关系模型的目录访问：通过提供数据访问的同一工具来访问结构和数据。
- ▶ 5. 全面的数据子语言规则：支持一种可以用于所有交互操作的语言（SQL就是从Codd的规则开发出来的）。
- ▶ 6. 视图更新规则：数据可以在不同的视图下呈现给用户，由用户查看、更新和删除。（当前系统在视图的更新和删除方面存在一定的限制）



Codd提出的关于关系数据库的12条规则

- ▶ 7. 高级的插入、更新和删除：增、删、改操作应该可以对任何一组数据进行，而非局限于一张表或表中的一行记录。
- ▶ 8. 物理结构独立：改变数据库的物理存储方法不会影响用户对数据的访问方式。
- ▶ 9. 逻辑数据独立：改变数据库的逻辑结构也不会影响用户对数据的访问。（目前系统在实现上有一定的困难）
- ▶ 10. 完整性独立：与数据库进行交互时使用的语言应该支持用户提出的约束，这样将能维护数据的完整性。
- ▶ 11. 分布独立：如果数据库是分布的（物理上分布于多个计算机上），这个情况用户应感觉不到。
- ▶ 12. 没有子版本规则：除了使用数据库语言以外应该不能用任何其他的方法来改变数据库结构。

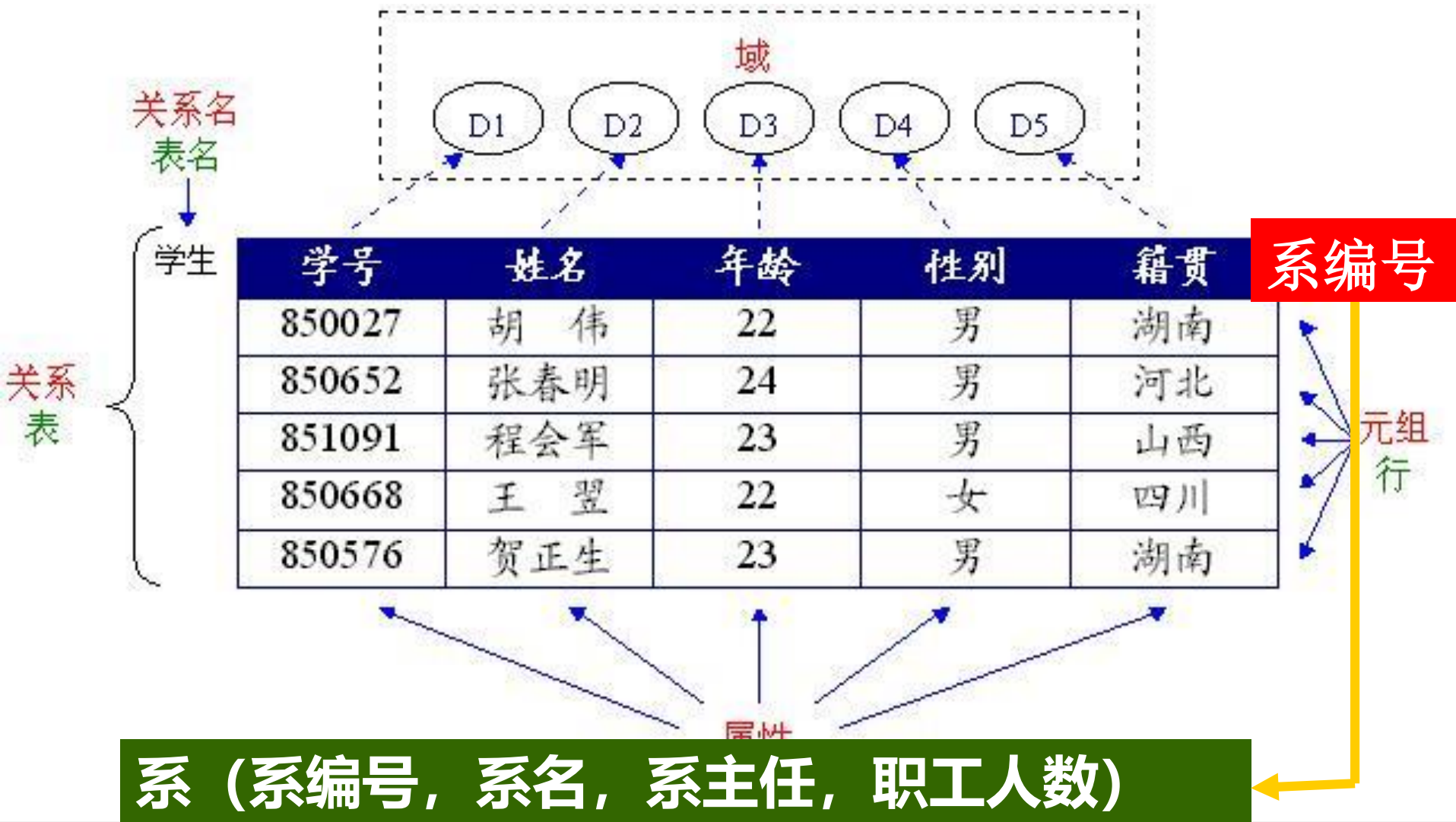


关系数据模型

用**二维表**来表示实体集，用
外键表示实体间的联系，这样的
数据模型称为**关系数据模型**



关系数据模型





关系模型的组成

数据结构：全部数据及相互联系都被组织成关系

数据操作：查询和更新

完整性规则：关系模型的三类完整性规则（实体完整性、参照完整性、用户定义的完整性）



本章内容

关系模型概述

关系数据结构及形式化定义

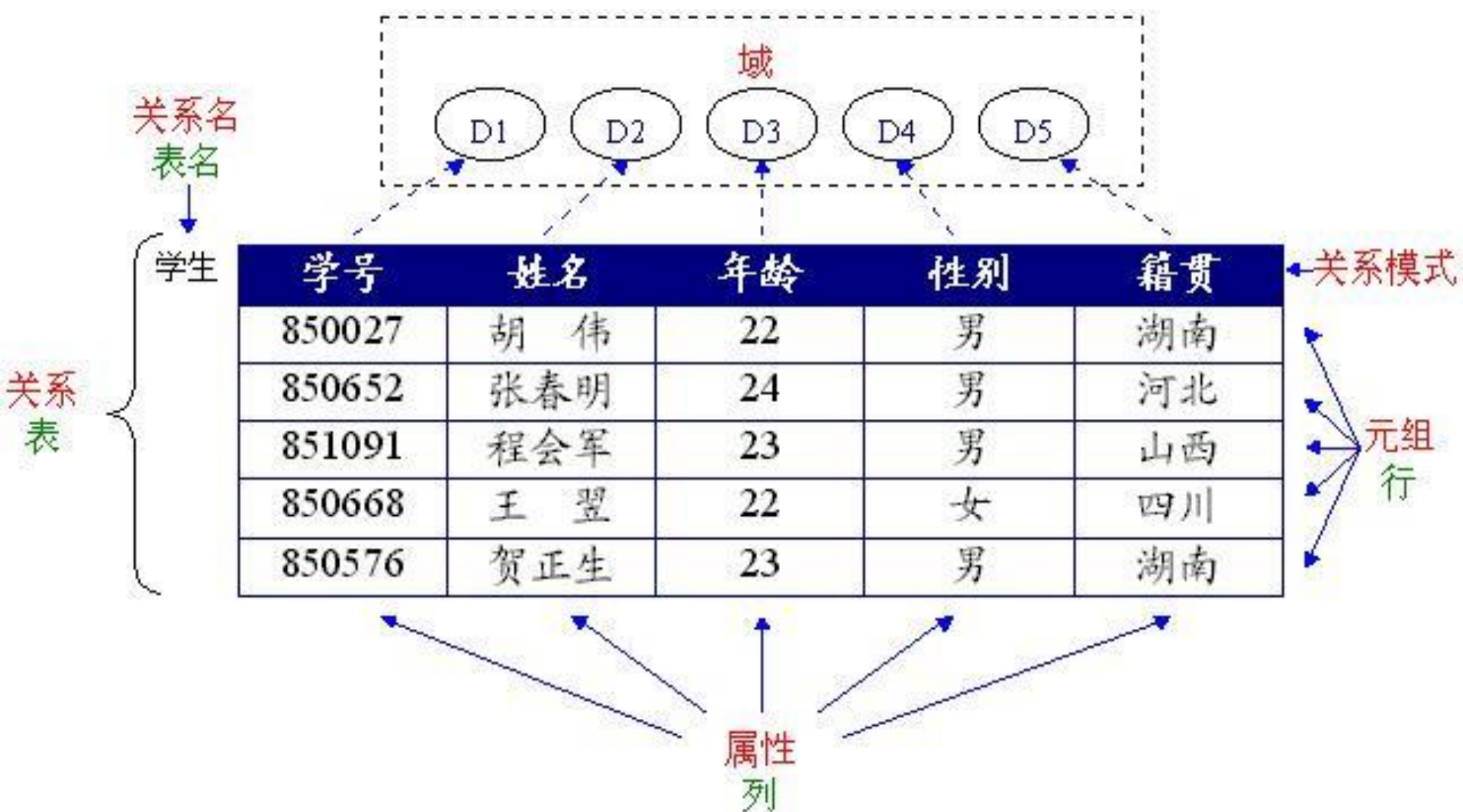
关系的完整性

关系代数



第一节 关系数据结构及形式化定义

- ▶ **1. 关系**
- ▶ **2. 关系模式**
- ▶ **3. 关系数据库**





1. 关系—Relation

联系：

contact; touch;
connection; link

► 单一的数据结构—关系

现实世界的实体以及实体间的各种联系均用关系来表示

► 逻辑结构—二维表

从用户角度，关系模型中数据的逻辑结构是一张二维表

► 建立在集合代数的基础上，algebra of sets

- 为什么是集合代数？什么叫代数？



关系概念的相关形式化定义

- 1) 域 (Domain)
- 2) 笛卡尔积 (Cartesian Product)
- 3) 关系 (Relation)



1) 域 (Domain)

► 域是一组具有**相同数据类型**的值的集合。例：

- 整数
- 实数
- 介于某个取值范围的整数
- 指定长度的字符串集合
- { '男' , '女' }
-



2) 笛卡尔积 (Cartesian Product)

► 笛卡尔积

给定一组域 D_1, D_2, \dots, D_n , 则

D_1, D_2, \dots, D_n 的笛卡尔积为:

$$D_1 \times D_2 \times \dots \times D_n =$$

$$\{ (d_1, d_2, \dots, d_n) \mid d_i \in D_i, i = 1, 2, \dots, n \}$$

- 所有域的所有取值的一个组合
- 不能重复

► D_1, \dots, D_n , 这些域中可以有**相同域**

René Descartes: was a French philosopher, mathematician, and scientist



3) 笛卡尔积 (Cartesian Product)

例如，给出三个域：

D_1 = 导师集合 = 张清玫，刘逸

D_2 = 专业集合 = 计算机专业，信息专业

D_3 = 研究生集合 = 李勇，刘晨，王敏

该笛卡尔积的**基数**为 $2*2*3=12$ 。一共有12个元组。



笛卡尔积的例子

D_1 , D_2 , D_3 的笛卡尔积

元组

分量

supervisor	speciality	postgraduate
张清玫	计算机专业	李勇
张清玫	计算机专业	刘晨
张清玫	计算机专业	王敏
张清玫	信息专业	李勇
张清玫	信息专业	刘晨
张清玫	信息专业	王敏
刘逸	计算机专业	李勇
刘逸	计算机专业	刘晨
刘逸	计算机专业	王敏
刘逸	信息专业	李勇
刘逸	信息专业	刘晨
刘逸	信息专业	王敏



笛卡尔积元组的实际意义

- 假设一个导师只有一个专业，一个研究生只能师从一个导师，一个导师可带多个研究生。则上述笛卡尔积中许多元组是没有实际意义的，取出有实际意义的元组来构造导师与研究生的关系，取名为SAP关系。

SAP 关系

supervisor	speciality	postgraduate
张清玫	信息专业	李勇
张清玫	信息专业	刘晨
刘逸	信息专业	王敏



关系 (Relation) 的定义

► (1) 关系

- $D_1 \times D_2 \times \dots \times D_n$ 的**子集**称作在域 D_1, D_2, \dots, D_n 上的关系, 表示为
- $R(D_1, D_2, \dots, D_n)$
- R : 关系名
- n : 关系的目或度 (Degree)
- n 元关系, 一元关系(Unary relation), 二元关系(Binary relation)



元组与二维表

(2) 元组—tuple

- 关系中的每个元素是关系中的元组，通常用t表示。

(3) 关系的表示

- 关系也是一个二维表，表的每行对应一个元组，表的每列对应一个域

SAP 关系

supervisor	speciality	postgraduate
张清玫	信息专业	李勇
张清玫	信息专业	刘晨
刘逸	信息专业	王敏



关系的属性及其类别

(4) 属性

- 关系中**不同列可以对应相同的域**
- 为了加以区分，必须对每列起一个名字，称为属性 (Attribute)
- n 目关系必有 n 个属性

问题：用什么值来唯一地代表一个元组？



特殊的属性集

► 候选码—candidate key

- 若关系中的**某一属性组**的值能**唯一地**标识一个元组，则称该属性组为候选码(键)

► 主码，主键—primary key

- 从候选码中由人为选定一个作为主码

► 外键(外码): foreign key

- 如果关系R的某一个或一组属性不是本身的码，而是另一关系S的主码，则称该属性或属性组是R的外键(外码)



案例

职工（职工编号，职工身份证号，姓名，部门编号，性别）

候选键（职工编号），（职工身份证号）

部门（部门编号，部门名称，部门经理）

候选键（部门编号）

主键（职工编号）

外键（部门编号）



属性类别

- ▶ **主属性: primary attribute**

候选码的诸属性

- ▶ **非主属性: non-prime attribute, non-key attribute**

不包含在任何候选码中的属性

- ▶ **全码:**

关系模式的所有属性是该关系模式的候选码, 称为全码

注意:

主属性是候选码中的属性, 候选码由主属性构成, 但主属性不一定是候选码



关系的性质

- 一个关系中**不存在**两个元组在各个分量（或属性）上完全相同；
- **行**的次序无所谓；
- **列**的次序无所谓；
- 每个分量必须是**不可分**的量；
- 列的分量是**同一类型**的数据，来自同一域。



最基本的规范化要求

关系模型要求**关系必须规范化**，即**满足一定的规范条件**。规范条件中**最基本**的一条是：**关系的每一个分量必须是一个不可分的数据项**。

非规范化关系

supervisor	speciality	postgraduate	
		PG1	PG2
张清玫	信息专业	李勇	刘晨
刘逸	信息专业	王敏	

这样的关系在关系数据库中不允许的。



规范化的关系

SAP 关系

supervisor	speciality	postgraduate
张清玫	信息专业	李勇
张清玫	信息专业	刘晨
刘逸	信息专业	王敏



请指出如下属性集的问题

- ▶ **联系人（昵称，微信号，头像，备注名，标签，电话号码，描述，附加图片，地区，个人相册，更多）**
- ▶ **哪些不是属性？**
- ▶ **哪些不符合规范要求？**
- ▶ **哪些不应该列入？**



行列次序可以任意交换

行的次序无所谓；
列的次序无所谓；

R_1

A_1	A_2
1	a
2	b
3	a

R_2

A_2	A_1
a	1
b	2
a	3

R_3

A_2	A_1
a	3
b	2
a	1



三类关系

关系有三类：**基本表**、**查询表**、**视图表**

基本表：实际存在的表，是实际存储数据的逻辑表示。

查询表：查询结果对应的表。

视图表：由基本表或其他视图表导出的**虚表**，不对应实际存储的数据。



2.关系模式—relation schema

关系是元组的集合，**关系模式**指出元组集合的**结构**，以及关系遵循的**完整性约束条件**。

关系模式是类型，一个具体的关系是对应的关系模式的一个值。

关系模式组成成分：

属性集 U ，域集 D ，属性与域之间来源关系集 DOM ，属性间的依赖关系集 F

关系模式的形式化表示：

$R(U, D, DOM, F)$

谁需要这么严谨复杂的信息？



关系模式简化表示

关系模式仅仅是对数据特性的描述。系模式定义至少应包括：**模式名、属性名、值域名和完整性约束。**

简化的说法：**二维表的表头那一行称为关系模式。**
又称**表的框架或记录类型。**

关系模式可以简记为：

关系模式名 (属性名₁, 属性名₂, ..., 属性名_n)

示例：学生 (学号, 姓名, 年龄, 性别, 籍贯)



关系模式与关系的区别

问题：

关系模式和关系有什么区别？

关系模式是**静态**的（型），关系是关系模式某一时刻的状态，是**动态**的（值）。

关系是不是一个集合？

关系什么的集合？



3. 关系数据库

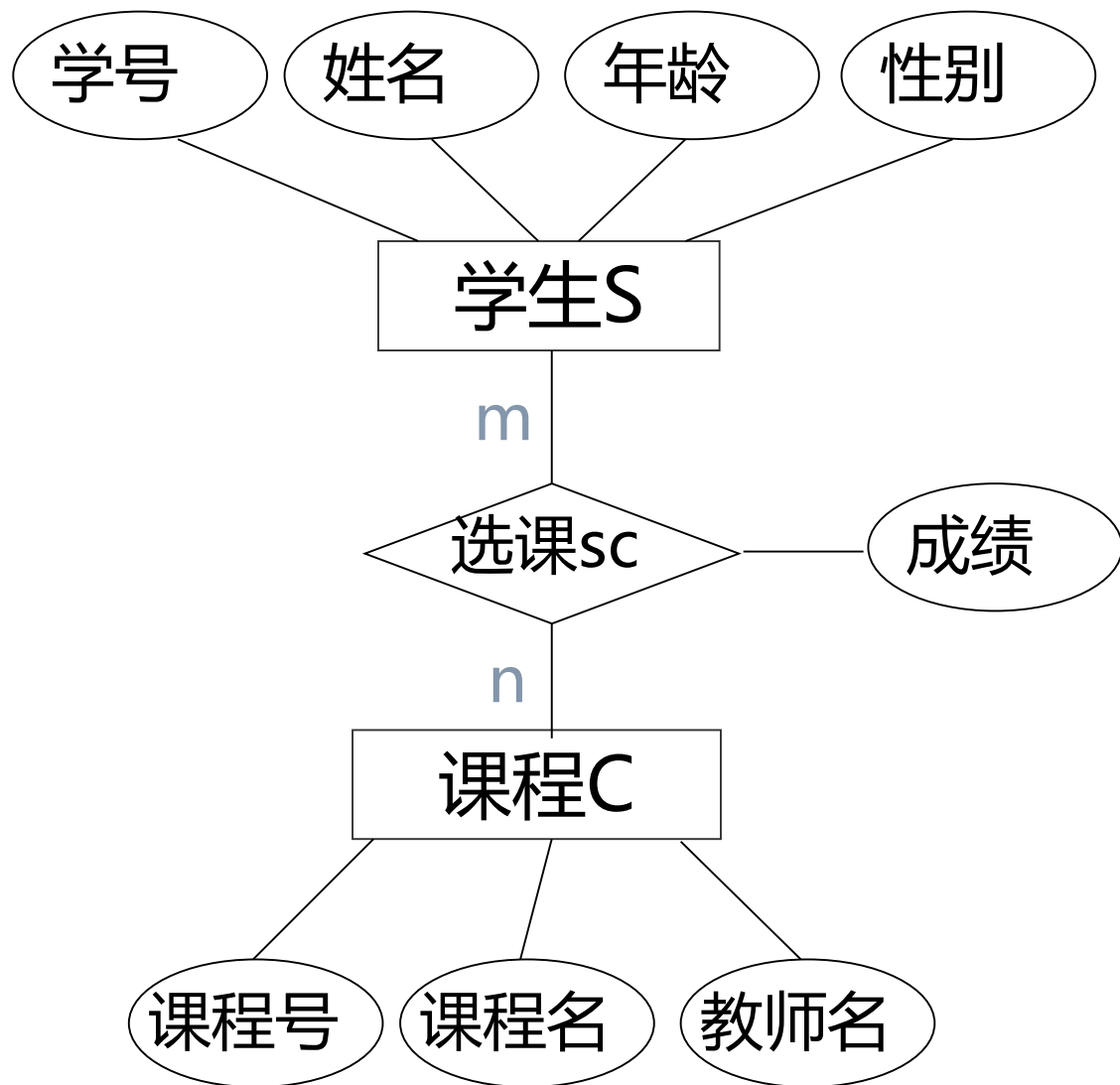
关系模型中，**实体**以及实体间的**联系**都用**关系**来表示。在一个给定的应用领域中，所有**关系的集合**构成一个**关系数据库**。

关系数据库模式是对整个**关系数据库**的**描述（型）**，**关系数据库**是**所有关系模式**某**一时**刻对应的**关系的集合（值）**。



例：学生选课概念模型

学号—S#
姓名—SN
年龄—age
性别—sex
课程号—C#
课程名—CN
教师名—T
成绩—G





学生选课关系模式集

学生关系模式 $S(\textcolor{red}{S\#}, \text{SN}, \text{age}, \text{sex})$

课程关系模式 $C(\textcolor{blue}{C\#}, \text{CN}, \text{T})$

选课关系模式 $SC(\textcolor{red}{S\#}, \textcolor{blue}{C\#}, \text{G})$

★在关系模式 SC 中 $(S\#, C\#)$ 是**候选键**，单个的 $S\#$ 或 $C\#$ 都不是，但它们分别是关系模式 S 和 C 中的候选键，故 $S\#$ 和 $C\#$ 称为 SC 的**外键**。



关系模式的实例→关系表

学生关系模式 $S(S\#, SN, age, sex)$ 的一个实例

S#	SN	age	sex
S1	LI	17	M
S2	WANG	19	F
S3	LIU	21	F
S4	CHEN	20	M



关系模式的实例→关系表

课程关系模式 $C(C\#, CN, T)$ 的一个实例

C#	CN	T
C1	MATHS	LIU
C2	PHYSICS	HUANG
C3	DB	SHI
C4	OS	ZHANG



关系模式的实例→关系表

选课关系模式 $SC(S\#, C\#, G)$ 的一个实例

S#	C#	G
S1	C1	90
S1	C2	78
S2	C2	89
S3	C3	75
S4	C4	80

返回



本章内容

关系模型概述

关系数据结构及形式化定义

关系的完整性

关系代数



第三节 关系的完整性

关系的完整性规则是对数据的**约束**

关系模型提供**三类**完整性规则：

实体完整性、参照完整性和用户定义的完整性

实体完整性

参照完整性

是关系模型必须满足的完整性约束条件，由关系系统自动支持

用户定义的完整性

—— 是应用领域需要遵循的约束条件，业务规则



关系模型的三类完整性规则

(1) 实体完整性规则

基本关系R的主属性**不能取空值**。

(2) 参照完整性规则

定义主--外码之间的引用规则。不引用不存在的实体。
若属性集K是关系模式R1的主键，K也是关系模式R2的外键，K的取值**或为空，或等于R1关系中的某个主键值**。R1为**被参照关系**（目标关系），R2为**参照关系**。



主码

参照完整性案例

学生 (学号, 姓名, 性别, 专业号, 年龄)

专业 (专业号, 专业名)

外码

主码

❖ 学生关系引用了专业关系的主码“专业号”, 学生关系是参照关系, 专业关系是被参照关系。

❖ 学生关系中每个元组的“专业号”属性只取两类值:

(1) 空值, 表示尚未给该学生分配专业

(2) 非空值, 这时该值必须是专业关系中某个元组的“专业号”值, 表示该学生不可能分配一个不存在的专业



参照完整性注意事项

1) **外键**和相应的**主键**可以**不同名**，只要定义在**相同的值域**上即可（不同关系中往往同名）

职工（职工号，职工名，年龄，**部门编号**）

部门（部门号，部门名，负责人）

2) 参照与被参照关系可以是**同一个关系**，表示了属性之间的联系。（**关系内部**的引用）如：

职工（职工号，姓名，年龄，部门号，**负责人职工号**）

也可以是两个**不同的关系**，表示实体之间的引用（**关系间**的引用）



参照完整性注意事项

3) 外键值是否允许空，应视具体问题而定。

例1: **SC(S#,C#,G)** 学生关系 $\xleftarrow{\text{学号}}$ 选修关系 $\xrightarrow{\text{课程号}}$ 课程关系
(b)

S#和C#是选修关系的主属性，不能取空值（实体完整性规则），只能取被参照关系中已经存在的主码值。

例2: **职工（职工号，姓名，年龄，部门号，负责人职工号）**

负责人职工号可以取空值（所在部门未有负责人）或非空值（本关系中某个元组的职工号）



(3) 用户定义的完整性规则

- ▶ 针对不同的应用环境而定义的约束条件。
- ▶ 关系模型提供定义和检验这类完整性的机制，不一定非用程序完成

例：课程(课程号，课程名，学分)

- “课程号” 属性必须取唯一值
- 非主属性 “课程名” 也不能取空值
- “学分” 属性只能取值{1, 2, 3, 4}



几种完整性案例

关系模式集

学生关系模式 $S(S\#, SN, age, sex)$

课程关系模式 $C(C\#, CN, T)$

选课关系模式 $SC(S\#, C\#, G)$

- 1) 在 $S(S\#, SN, age, sex)$ 中，主属性 $S\#$ 不能为空值。在 $C(C\#, CN, T)$ 关系中，主属性 $C\#$ 不能为空值。
- 2) 在 $SC(S\#, C\#, G)$ 中，主属性 $S\#$ 和 $C\#$ 是外码，不能为空值，且 $S\#$ 只能取 $S(S\#, SN, age, sex)$ 中的 $S\#$ 值， $C\#$ 只能取 $C(C\#, CN, T)$ 中的 $C\#$ 值
- 3) 若规定学生年龄在18到35之间，则是用户定义的完整性规则。



关系模型的形式定义

数据结构：全部数据及相互联系都被组织成关系

数据操作：关系运算 { **关系代数**
关系演算

完整性规则：关系模型的三类完整性规则



本章内容

关系模型概述

关系数据结构及形式化定义

关系的完整性

关系代数



第四节 关系代数

关系数据库的数据操作分为**查询**和**更新**

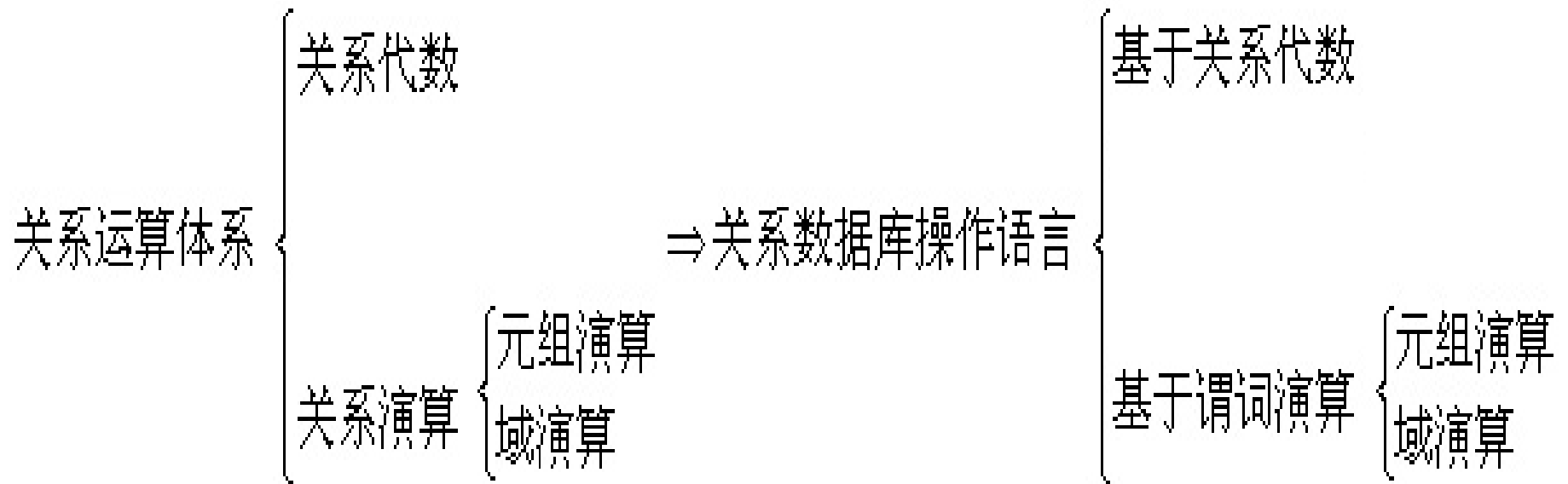
关系查询语言根据其理论基础不同分成两大类：

- **关系代数语言**：是用**关系（集合）运算**来表达查询要求的DML语言
- **关系演算语言**：用**谓词演算**来表达查询要求的DML语言

★各种关系查询语言均属于“非过程性”语言，关系代数语言的非过程性较弱。



关系代数



任何一种运算都是将一定的**运算符**作用于一定的**运算对象**，得到预期的**运算结果**。**关系代数的运算对象和结果都是关系。**

运算符包括**四类**：集合运算符、专门关系运算符、算术比较符、逻辑运算符

关系代数运算符

运算符		含义	运算符		含义
集 合 运算符	\cup	并	比 较 运算符	$>$	大于
	$-$	差		\geq	大于等于
	\cap	交		$<$	小于
	\times	广义笛卡尔积		\leq	小于等于
专门的 关 系 运算符	σ	选择		$=$	等于
	π	投影	逻 辑 运算符	\neq	不等于
	\bowtie	连接		\neg	非
	\div	除		\wedge	与
				\vee	或



(1) 关系代数的定义

关系代数是一组建立在关系上的高级运算，每个运算都以一个或多个关系作为它的运算对象，并且**生成一个关系**作为运算结果。

运算对象：关系

运算结果：关系



(2) 关系运算的类别

关系运算分两类:

(a) 传统的集合运算:

合并 \cup 、相交 \cap 、求差- (相减)、笛卡尔积 \times

(运算从关系的水平 (行) 的角度来进行)

(b) 专门的关系运算:

选择 σ 、投影 π 、联接 \bowtie 、求商 \div
(运算不仅涉及行而且涉及列)



① 合并

两个关系 R_1, R_2 （**属性数相同，且相应属性取自同一个域**）的合并，是由属于 R_1 或属于 R_2 （或属于两者）的所有元组 t （**不计重复元组**）组成的一个新的关系。

运算符：“ \cup ” 记为： $R_1 \cup R_2$

$$R_1 \cup R_2 = \{t \mid t \in R_1 \vee t \in R_2\}$$



例如

R

A	B	C
1	2	3
4	5	6
7	8	9

S

A	B	C
7	8	9
4	5	6
5	1	12

$R \cup S$ 的结果为:

A	B	C
1	2	3
4	5	6
7	8	9
5	1	12



② 差（相减）

两个关系 R_1, R_2 （属性数相同，且相应属性取自同一个域）的求差，是由属于 R_1 而不属于 R_2 的所有元组 t 组成的一个新的关系。

运算符：“-” 记为： $R_1 - R_2$

$$R_1 - R_2 = \{t \mid t \in R_1 \wedge t \notin R_2\}$$



例如

R

A	B	C
1	2	3
4	5	6
7	8	9

S

A	B	C
7	8	9
4	5	6
5	1	12

R-S的结果为:

A	B	C
1	2	3



③ 笛卡尔积

两个关系R,S (设R为 k_1 元关系, M 个元组, S为 k_2 元关系, N 个元组) 的广义笛卡尔积, 是一个 (K_1+K_2) 列的元组的集合, 是由属于R的任何一个元组 t^{k1} 和属于S的元组 t^{k2} 连接而成的新元组 t 所组成的一个新关系。

(新关系中元组的前 K_1 列是关系R的一个元组, 后 K_2 列是关系S的一个元组, 基数为 $M*N$)

$$R \times S = \{t \mid t = \langle t^{k_1}, t^{k_2} \rangle \wedge t^{k_1} \in R \wedge t^{k_2} \in S\}$$



例如

$$D = \begin{bmatrix} 1 \\ 2 \\ 3 \end{bmatrix} \times \begin{bmatrix} a \\ b \end{bmatrix} \Rightarrow D = \begin{bmatrix} 1 & a \\ 1 & b \\ 2 & a \\ 2 & b \\ 3 & a \\ 3 & b \end{bmatrix}$$



例如： $R \times S$

R

A	B	C
a	b	c
b	c	e
e	d	c

S

C	D
c	d
e	f

$R \times S$ 结果为：

$R \times S$ 结果中元组的个数 (即基数, 行数)
= R和S中行数之积;
属性的个数 (即元数, 列数)
= R和S中列数之和。

A	B	R.C	S.C	D
a	b	c	c	d
a	b	c	e	f
b	c	e	c	d
b	c	e	e	f
e	d	c	c	d
e	d	c	e	f

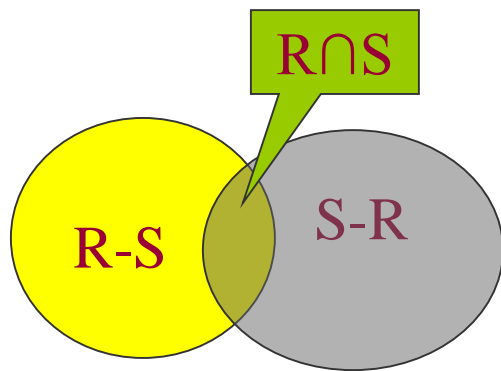


④ 交

两个关系 R_1, R_2 (**属性数相同，且相应属性取自同一个域**) 的相交，是由既属于 R_1 又属于 R_2 的所有元组 t 组成的一个新的关系。

运算符：" \cap " **记为：** $R_1 \cap R_2$

$$R_1 \cap R_2 = \{t \mid t \in R_1 \wedge t \in R_2\} \Leftrightarrow R_1 \cap R_2 = R_1 - (R_1 - R_2)$$



$$\begin{aligned} R \cap S &= R - (R - S) \\ R \cap S &= S - (S - R) \end{aligned}$$



例如: $R \cap S$

例如

R

A	B	C
1	2	3
4	5	6
7	8	9

S

A	B	C
7	8	9
4	5	6
5	1	12

$R \cap S$ 的结果为:

A	B	C
4	5	6
7	8	9



⑤ 选择（限制）

从现有关系中选择满足一定条件的**元组**组成新的关系。（从行的角度进行运算）

运算符 “ δ ” 记为：

$$\delta_F(R) = \{t | t \in R \wedge F(t) = \text{'真'}\}$$

其中**F**为**选择条件**，是一个逻辑表达式——

运算对象：常量、属性名(序号)、简单函数

运算符：算术比较运算符、逻辑运算符



例如

$$\delta_{A_2 > 4 \wedge A_1 = 'a'}(R) = \{t \mid t \in R \wedge A_2 > 4 \wedge A_1 = 'a'\}$$

A_1	A_2	A_3
a	7	e
a	2	f
a	5	g
b	4	h

结果

A_1	A_2	A_3
a	7	e
a	5	g



⑥ 投影

从现有关系中**选取某些属性** (列),组成新的关系。**(从列的角度进行运算)**

若 R 是一个 k 元关系, 其元组变量为 $t^k = \langle t_1, t_2, \dots, t_k \rangle$; 那么关系 R 在其分量 $t_{j_1}, t_{j_2}, \dots, t_{j_n}$ ($n \leq k$; j_1, j_2, \dots, j_n 为 1 到 k 之间互不相同的整数) 上的投影(**运算符 “ π ”**) 记为:

$$\pi_{j_1, j_2, \dots, j_n}(R) = [t \mid t = \langle t_{j_1}, t_{j_2}, \dots, t_{j_n} \rangle \wedge \langle t_{j_1}, t_{j_2}, \dots, t_{j_n} \rangle \in R]$$



例如

$$\pi_{A_3, A_1}(R) = \{t \mid t = \langle A_3, A_1 \rangle \wedge \langle A_1, A_2, A_3 \rangle \in R\}$$

R

A ₁	A ₂	A ₃
a ₁	b ₁	c ₁
a ₁	b ₂	c ₁
a ₂	b ₂	c ₂

结果:

A ₃	A ₁
c ₁	a ₁
c ₂	a ₂

例：查询所有年龄大于18岁的女生的学号和姓名。

$$\pi_{S\#, SN}(\delta_{AGE > 18 \wedge SEX = '女'}(S))$$



⑦连接 (θ - 连接)

连接也称为 θ 连接，它是从两个关系的笛卡尔积中选取属性间满足一定条件的元组。记作：

$$R \bowtie_{i \theta j} S = \sigma_{i \theta (r+j)} (R \times S)$$

其中 i 和 j 分别为 R 和 S 上第 i 个，第 j 个**属性的序号**。 θ 是**比较运算符**。连接运算从 R 和 S 的广义笛卡尔积 $R \times S$ 中选取第 i 个分量与第 $r+j$ 个分量满足 θ 运算的元组， r 为 R 中属性数。

i 和 j 也可以是属性组。



例如F连接

θ 可以是比较运算符中的任何一个

$<, >, \leq, \geq, =, \neq$

F_连接: 当F为公式时称为F_连接, 记为:

$$R \bowtie_F S$$

其中, F 为: $F_1 \wedge F_2 \wedge \dots \wedge F_n$

每个 F_i 都是形如 $i \theta j$ 的式子

等值连接: 所有公式中 θ 均为 $=$ 号



例如 θ 连接

θ -连接的例子：（只有一个运算条件的连接） （从行的角度进行运算）

R_1

A_1	A_2	A_3
a	4	e
b	2	f
c	5	g
d	1	h

R_2

A_1	A_2	A_3
b	2	e
c	3	f
b	5	h

求： $R_1 \bowtie_{2>2} R_2$



θ 连接结果

结果为：

$R_1.A_1$	$R_1.A_2$	$R_1.A_3$	$R_2.A_1$	$R_2.A_2$	$R_2.A_3$
a	4	e	b	2	e
a	4	e	c	3	f
c	5	g	b	2	e
c	5	g	c	3	f



F连接的例子

求

$$R \bowtie S$$

$R. A > S. E \wedge R. B < S. F$

结果为:

R

A	B	C
1	2	3
4	15	6
7	8	9

S

D	E	F
3	1	4
6	2	13
5	1	12

A	B	C	D	E	F
7	8	9	6	2	13
7	8	9	5	1	12



自然连接

自然连接是一种特殊的等值连接。要求两个关系中进行比较的分量是相同的属性组，并在结果中去掉重复的属性列。(从行和列的角度进行运算) 若关系R和S具有相同的属性组B，则自然连接记作为：

$$R \bowtie S = \{ \overbrace{t_r t_s} \mid t_r \in R \wedge t_s \in S \wedge t_r[B] = t_s[B] \}$$



例如

R

A ₁	A ₂	A ₃
a	1	b
b	2	c
c	4	e

S

A ₁	A ₂
e	2
a	1
b	3
c	4

R ⋈ S	A1	A2	A3
结果	a	1	b
	c	4	e



与等值连接的区别

(1)在做自然连接的两个关系中，要求值相等的属性名也必须相等，而在等值连接中不要求值相等的属性名相等。

(2)在自然连接的结果中，去掉重复的属性名，而在等值连接的结果中不去掉重复的属性名。

★若两个关系没有公共属性，则其自然连接就转化为笛卡尔积操作。



与等值连接的区别

R

A	B	C
a1	b1	5
a1	b2	6
a2	b3	8
a2	b4	12

S

B	E
b1	3
b2	7
b3	10
b3	2
b5	2

等值连接 $R \bowtie_{2=1} S$

A	R.B	C	S.B	E
a1	b1	5	b1	3
a1	b2	6	b2	7
a2	b3	8	b3	10
a2	b3	8	b3	2

自然连接 $R \bowtie S$

A	B	C	E
a1	b1	5	3
a1	b2	6	7
a2	b3	8	10
a2	b3	8	2



象集—Image Set

给定一个关系 $R(X, Y)$, X 和 Y 为属性组。定义, 当 $t[X]=x$ 时, x 在 R 中的象集为:

$$Y_x = \{t[Y] \mid t \in R, t[X]=x\}$$

它表示 R 中属性组 X 上值为 x 的诸元组在 Y 上各分量的集合。

$t[Y]$

Y 为 R 中的一个属性组, $Y = \{y_{i1}, y_{i2} \dots y_{ik}\}$,

$t[Y] = (t[y_{i1}], t[y_{i2}] \dots t[y_{ik}])$ 表示 R 中的元组 t 在属性组 Y 上各分量的集合



例如

R: X Y

A	B	C	D
a	b	c	d
a	b	e	f
a	b	d	e
b	c	e	f
e	d	c	d
e	d	e	f

当 $x=(a,b)$ 时，其象集 Y_x 为：

C	D
c	d
e	f
d	e

当 $x=(b,c)$ 时，其象集 Y_x 为：

C	D
e	f

当 $x=(e,d)$ 时，其象集 Y_x 为：

C	D
c	d
e	f



⑧ 除法 (÷)

给定关系R (X, Y) 和S (Y, Z) , 其中X, Y, Z为属性组。R中的Y与S中的Y可以有不同的属性名, 但必须出自相同的值域。R与S的除运算得到一个新的关系P

(X) , P是R中满足下列条件的元组在X属性列上的投影; 元组在X上分量值x的象集 Y_x 包含S在Y上投影的集合。(从行和列的角度进行运算) 记为:

$R \div S = \{t_r[X] \mid t_r \in R \wedge \pi_y(S) \subseteq Y_x\}$, 其中 Y_x 为x在R中的象集,
 $x = t_r[X]$



例如

R: X Y

A	B	C	D
a	b	c	d
a	b	e	f
a	b	d	e
b	c	e	f
e	d	c	d
e	d	e	f

S: Y

C	D
c	d
e	f

当 $x=(a,b)$ 时, 其象集 Y_x 为:

当 $x=(b,c)$ 时, 其象集 Y_x 为:

C	D
e	f

当 $x=(e,d)$ 时, 其象集 Y_x 为:

C	D
c	d
e	f

C	D
c	d
e	f
d	e

$R \div S$ 的结果为:
X

A	B
a	b
e	d



例如

例: $R \div S$

R

X		Y	
A	B	C	
a_1	b_1	c_2	
a_2	b_3	c_7	
a_3	b_4	c_6	
a_1	b_2	c_3	
a_4	b_6	c_6	
a_2	b_2	c_3	
a_1	b_2	c_1	

S

Y		Z
B	C	D
b_1	c_2	d_1
b_2	c_1	d_1
b_2	c_3	d_2

(1) 当 $x = (a_1)$ 时
其象集为

B	C
b_1	c_2
b_2	c_3
b_2	c_1

X 可以取值为 $\{a_1, a_2, a_3, a_4\}$



例如

(2) 当 $x = (a_2)$ 时
其象集为

B	C
b_3	c_7
b_2	c_3

S 在 Y(B, C) 上的投影为

B	C
b_1	c_2
b_2	c_1
b_2	c_3

(3) 当 $x = (a_3)$ 时
其象集为

B	C
b_4	c_6

(4) 当 $x = (a_4)$ 时
其象集为

B	C
b_6	c_6

$R \div S$ 的结果为:

x

A
a_1



$R \div S$ 的操作步骤

(1) 将R中属性分为两个集合X和Y,

$R(X, Y)$, 其中Y是R和S中具有相同 值域的属性集合, $S(Y, Z)$ 。

(2) 若X的某个值x的象集 Y_x

$$Y_x = \{t[Y] \mid t \in R \wedge t[X] = x\}$$

包含S表中t[Y]的所有元组, 则
将x放入结果集中。



R 能被 S 除的充分必要条件

R 中包含 S 中的部分属性，（R 与 S 中的属性可以不同名，但必须有相同的值域） R 中有一些属性不出现在 S 中。



例7 检索选修全部课程的学生学号

$$\pi_{S\#, C\#}(SC) \div \pi_{C\#}(C)$$

S#	C#	G
S1	C1	90
S1	C2	78
S1	C3	86
S1	C4	77
S2	C1	76
S2	C2	89
S3	C3	75
S4	C4	80

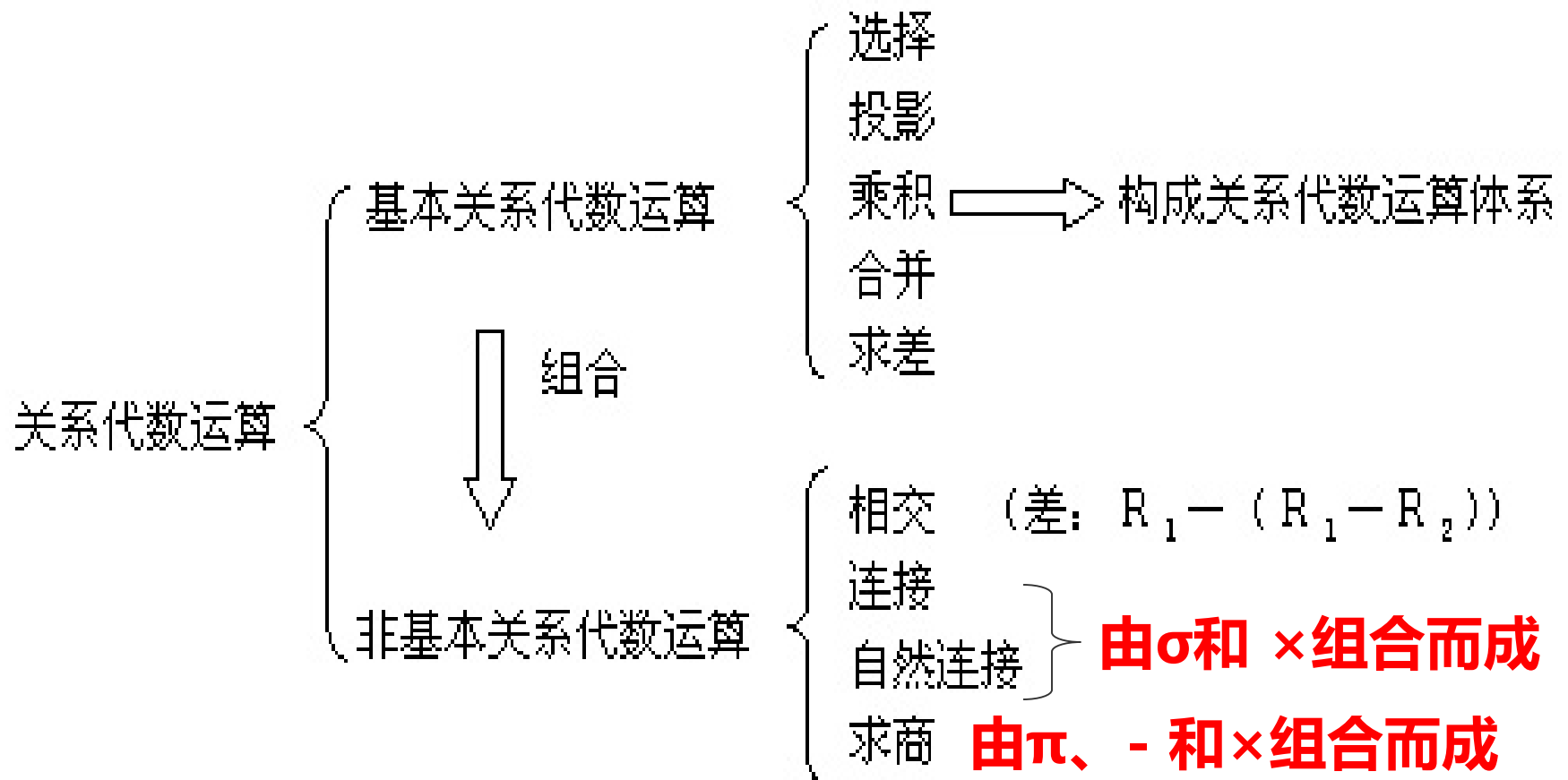
C#	CN	T
C1	MATHS	LIU
C2	PHYSICS	LI
C3	DB	SHI
C4	OS	FAN



S#
S1



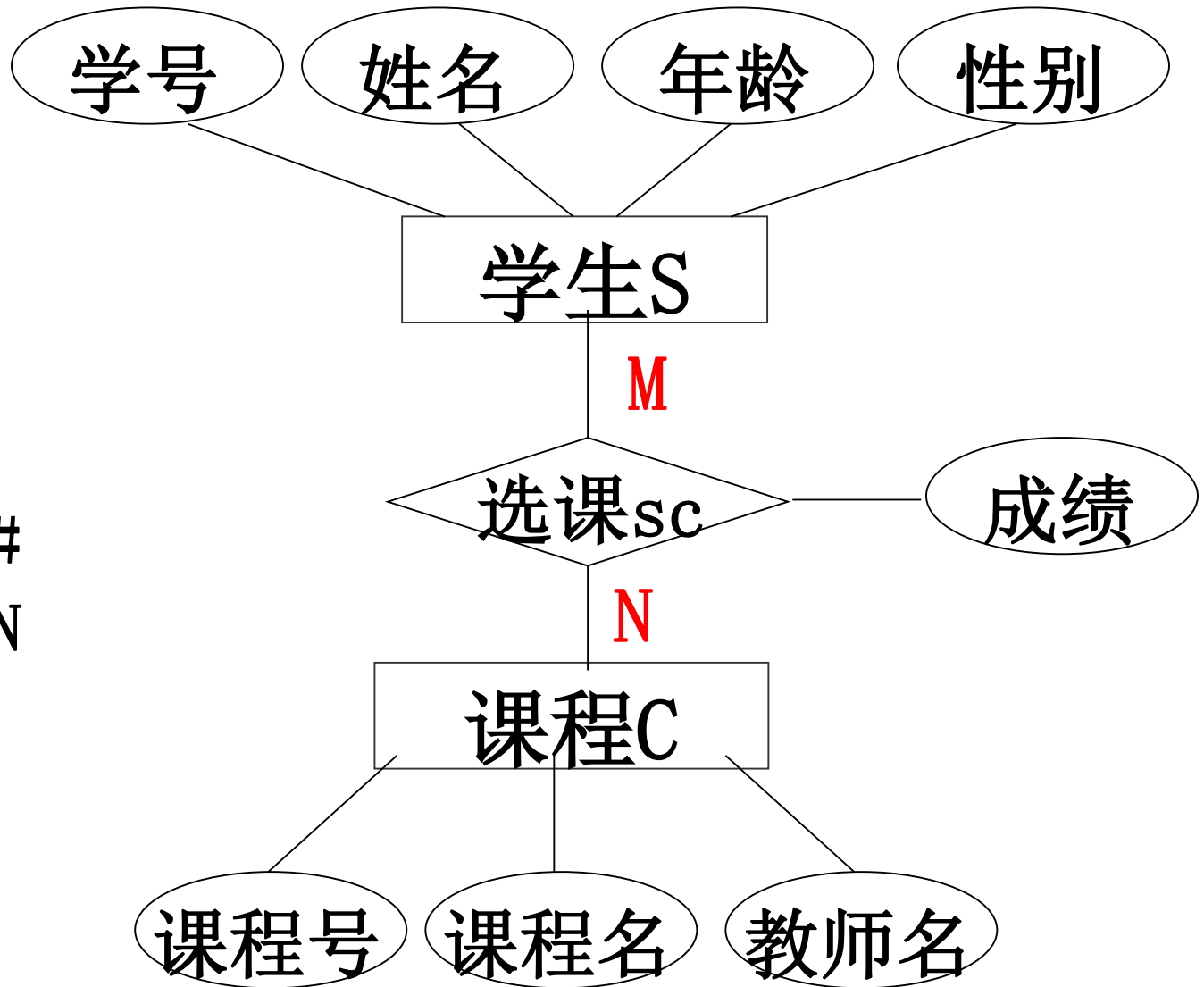
关系代数操作





关系代数应用实例：

学号—S#
姓名—SN
年龄—age
性别—sex
课程号—C#
课程名—CN
教师名—T
成绩—G





关系模式集

学生关系模式 $S(S\#, SN, age, sex)$

课程关系模式 $C(C\#, CN, T)$

选课关系模式 $SC(S\#, C\#, G)$



学生关系模式

学生关系模式 $S(\text{S\#}, \text{SN}, \text{age}, \text{sex})$ 的一个实例

S#	SN	age	sex
S1	LI	17	M
S2	SHI	19	F
S3	LIU	21	F
S4	CHEN	20	M



课程关系模式

课程关系模式 $C(C\#, CN, T)$ 的一个实例

C#	CN	T
C1	MATHS	LIU
C2	PHYSICS	LI
C3	DB	SHI
C4	OS	FAN



选课关系模式

选课关系模式 $SC(S\#, C\#, G)$ 的一个实例

S#	C#	G
S1	C1	90
S1	C2	78
S1	C3	86
S1	C4	77
S2	C1	76
S2	C2	89
S3	C3	75
S4	C4	80



练习

1. 检索学习课程号为C2的学生学号和成绩

$$\pi_{S\#, G}(\delta_{C\#='C2'}(SC))$$

$$\delta_{C\#='C2'}(\pi_{S\#, G}(SC))$$

错误!

2. 检索学习课程号为C2的学生学号和姓名

$$\pi_{S\#, SN}(\delta_{C\#='C2'}(S \bowtie SC))$$

$$\pi_{S\#, SN}(S \bowtie \delta_{C\#='C2'}(SC))$$

$$\pi_{S.S\#, SN}(\delta_{C\#='C2' \wedge S.S\#=SC.S\#}(S \times SC))$$



练习

3.检索选修课程名为DB的学生学号和姓名

$$\pi_{S\#, SN} (\delta_{CN='DB'} (S \bowtie SC \bowtie C))$$

$$\pi_{S.S\#, SN} (\delta_{CN='DB' \wedge S.S\#=SC.S\# \wedge SC.C\#=C.C\#} (S \times SC \times C))$$

4.检索选修课程号为C2或C4的学生学号

$$\pi_{S\#} (\delta_{C\#='C2' \vee C\#='C4'} (SC))$$



练习

5.检索至少选修课程号为C2和C4的学生学号

$$\pi_{S\#,C\#}(SC) \div \delta_{C\#='C2' \vee C\#='C4'}(C)$$

$$\pi_1(\delta_{1=4 \wedge 2='C2' \wedge 5='C4'}(SC \times SC))$$

1	2	3	4	5	6
S#	C#	G	S#	C#	G

6.检索不学课程号为C2的学生姓名和年龄

$$\pi_{SN,age}(S) - \pi_{SN,age}(\delta_{C\#='C2'}(S \bowtie SC))$$

$$\pi_{SN,age}(S \bowtie \delta_{C\# \neq 'C2'}(SC)) \cdot \cdot$$

错误!



选课关系模式 $SC(S\#,C\#,G)$ 的一个实例

S#	C#	G
S1	C1	90
S1	C2	78
S1	C3	86
S1	C4	77
S2	C1	76
S2	C2	89
S3	C3	75
S4	C4	80



练习

7.检索学习全部课程的学生姓名

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

8.检索所学课程包含学生S2所学课程的学生学号

$$\pi_{S\#, C\#}(SC) \div \pi_{C\#}(\delta_{S\#='S2'}(SC))$$



练习

例7 $\pi_{SN}(S \bowtie (\pi_{S\#, C\#}(SC) \div \pi_{C\#}(C)))$

S#	C#	G
S1	C1	90
S1	C2	78
S1	C3	86
S1	C4	77
S2	C1	76
S2	C2	89
S3	C3	75
S4	C4	80

C#	CN	T
C1	MATHS	LIU
C2	PHYSICS	LI
C3	DB	SHI
C4	OS	FAN



SN
LI

S#	SN	age	sex
S1	LI	17	M
S2	SHI	19	F
S3	LIU	21	F
S4	CHEN	20	M




练习

例8 检索所学课程包含学生S2所学课程的学生学号;

$$\pi_{S\#, C\#}(SC) \div \pi_{C\#}(\delta_{S\#='S2'}(SC))$$

X		Y
S#	C#	G
S1	C1	90
S1	C2	78
S1	C3	86
S1	C4	77
S2	C1	76
S2	C2	89
S3	C3	75
S4	C4	80

Y		
S#	C#	G
S2	C1	76
S2	C2	89



S#
S1
S2



练习

例9：检索选修了LIU老师所教的所有课程的学生号

$$\pi_{S\#, C\#} (SC) \div \pi_{C\#} (\delta_{T='LIU'} (C))$$

X S#	Y C#	G
S1	C1	90
S1	C2	78
S1	C3	86
S1	C4	77
S2	C1	76
S2	C2	89
S3	C3	75
S4	C4	80

Y C#	CN	T
C1	MATHS	LIU
C2	PHYSICS	LIU
C3	DB	SHI
C4	OS	FAN



S#
S1
S2



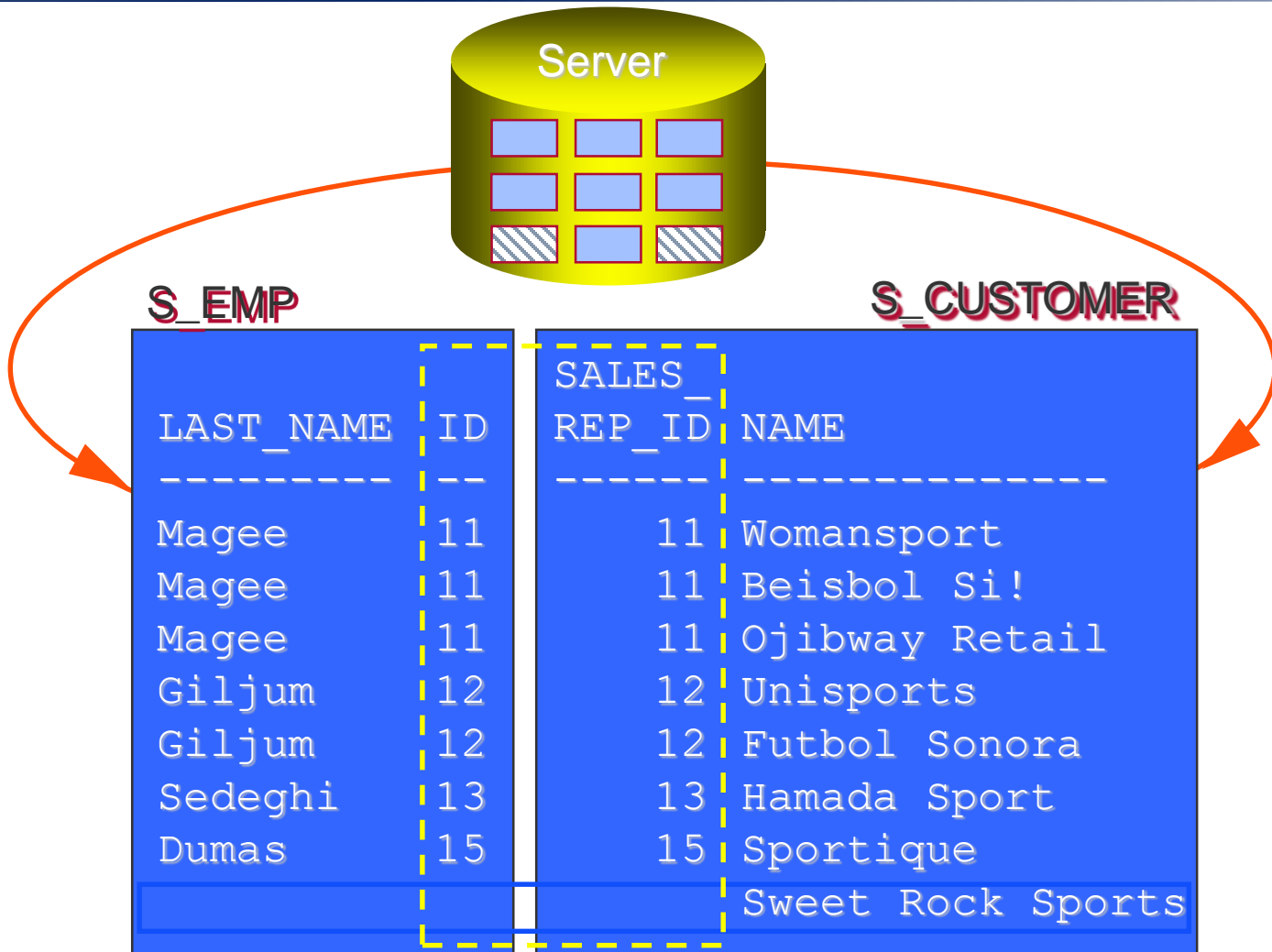
扩充的关系代数操作

在关系R和S做**自然联接**时，我们选择两个关系在公共属性上值相等的元组构成新关系的元组。

关系R中某些元组有可能在S中**不存在**公共属性上值相等的元组，造成R中的这些元组的值在操作时被**舍弃**。由于同样的原因。S中某些元组也有可能被舍弃。



外连接





外连接

如果关系R和S做**自然联接**时，把**R和S中原该舍弃的元组也保留**在新关系中，同时在这些元组**新增加的属性上填上空值（Null）**，这种操作称为“**外联接**”操作。用如下符号表示：

$$R \bowtie S$$



外连接的例子

R

A	B	C
a	b	c
b	b	f
c	a	d

S

B	C	D
b	c	d
b	c	e
a	d	b
e	f	g

$R \bowtie S$

A	B	C	D
a	b	c	d
a	b	c	e
c	a	d	b



外连接的例子

$R \bowtie S$

A	B	C	D
a	b	c	d
a	b	c	e
c	a	d	b
b	b	f	null
null	e	f	g



左外连接—保留左边，右边填空

如果关系**R**和**S**做**自然联接**时，只把**R**中原**该舍弃**的元组**放到新关系**中，那么这种操作称为“**左外联接**”操作。用如下符号表示：

$$R \bowtie S$$



左外连接的例子

R

A	B	C
a	b	c
b	b	f
c	a	d

S

B	C	D
b	c	d
b	c	e
a	d	b
e	f	g

$R \bowtie S$

A	B	C	D
a	b	c	d
a	b	c	e
c	a	d	b



左外连接的例子

$R \bowtie S$

A	B	C	D
a	b	c	d
a	b	c	e
c	a	d	b
b	b	f	null
null	e	f	g

$R \bowtie S$

A	B	C	D
a	b	c	d
a	b	c	e
c	a	d	b
b	b	f	null



右外连接—保留右边，左边填空

如果关系**R**和**S**做**自然联接**时，
只把**S**中原**该舍弃**的元组**放到新关系**
中，那么这种操作称为“**右外联接**”
操作。用如下符号表示：

$$R \bowtie \sqsupset S$$



右外联接的例子

R

A	B	C
a	b	c
b	b	f
c	a	d

S

B	C	D
b	c	d
b	c	e
a	d	b
e	f	g

$R \bowtie S$

A	B	C	D
a	b	c	d
a	b	c	e
c	a	d	b



右外联接的例子

$R \bowtie S$

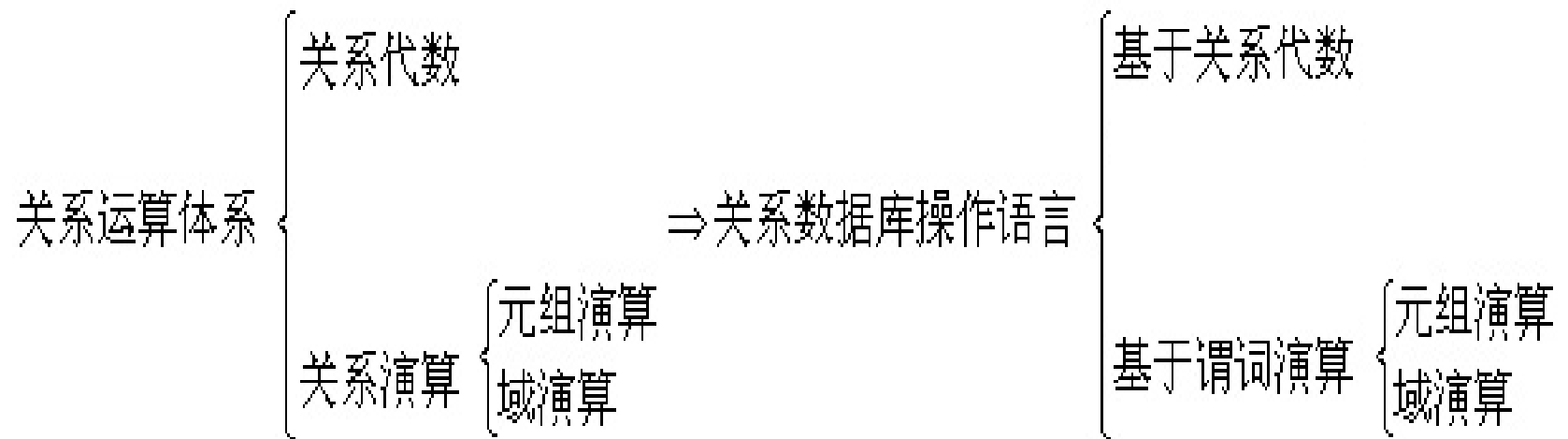
A	B	C	D
a	b	c	d
a	b	c	e
c	a	d	b
b	b	f	null
null	e	f	g

$R \bowtie S$

A	B	C	D
a	b	c	d
a	b	c	e
c	a	d	b
null	e	f	g



关系运算体系





本章重点

一、关系模型的基本概念

1.键: 1) 候选键(码)

2)主键(主码) 3)外键(外码)

2.关系模式的三类完整性规则

(1) 实体完整性规则

(2) 参照完整性规则

(3) 用户定义的完整性规则



本章重点

二、关系代数

1.传统的集合运算:

合并 \cup 、 相交 \cap 、 求差 $-$ (相减)、 笛卡尔积 \times

2.扩充的关系运算:

选择 σ 、 投影 π 、 联接 \bowtie 、 求商 \div

五个基本操作: \cup $-$ \times σ π



本章重点

3.关系代数表达式

- 关系代数运算经有限次复合后形成的式子
- 写关系代数表达式的步骤和注意事项
 - (1) 首先分析查询涉及几张**表**;
 - (2) 找出查询**条件**;
 - (3) 确定感兴趣的**数据项**;
 - (4) 如果涉及“**全部**”查询要用**除法**运算, 如果涉及“**否定**”查询要用**减法**运算;
 - (5) 最后将上述全部信息**组合**成代数表达式;
 - (6) 注意正确的关系运算**顺序**;
 - (7) 同一查询可用不同表达式实现, 应尽量采用**简单**和**效率高**的写法。



作业

- 第四版

- P.74

4, 5(仅用关系代数完成查询)

- 第五版

- P.70

5, 6(仅用关系代数完成查询)