

数据库原理与设计复习课

2013.1.8

试卷题型

- 选择题（15分）
- 填空题（15分）
- 简答题（5分 \times 6=30分）
- SQL查询(3分 \times 5=15分)
- 理论题（5分 \times 3=15分）//1题，但是3小题
- 设计题（10分）

本学期讲授的内容

- 第一章 绪论
- 第二章 关系数据库
- 第三章 关系数据库标准语言SQL
- 第四章 数据库安全性
- 第五章 数据库完整性
- 第六章 关系数据理论数据库设计
- 第七章 数据库设计
- 第十章 数据库恢复技术
- 第十一章 并发控制

第一章 绪论

- 数据库技术产生于六十年代末，是数据管理的最新技术，是计算机科学的重要分支。
- 经历了三代演变：层次/网状系统、关系系统、新一代数据库系统家族

§ 1.1 数据库技术发展历史

- 数据管理技术的发展过程

在应用需求的推动下，在计算机硬件、软件发展的基础上，数据管理技术经历了下面三个阶段：

- 人工管理阶段(40年代中--50年代中)
- 文件系统阶段(50年代末--60年代中)
- 数据库系统阶段(60年代末--现在)

第一章 绪论

1.1 数据库系统概述

1.1.1 四个基本概念

1.1.2 数据管理技术的产生和发展

1.1.3 数据库系统的特点

1.1.1 四个基本概念

- 数据(Data)
- 数据库(Database)
- 数据库管理系统(DBMS)
- 数据库系统(DBS)

一、数据

- 信息：客观世界中事物的存在方式和运动状态及其变化的反映，是客观事物之间相互联系和相互作用的表征。

一、数据


- 数据是对信息的符号化表示，即用一定的符号（数字、文字、图形、图象、声音等）来表示信息。
- 数据与信息的联系：数据是信息的载体，信息是数据的内涵。同一信息可以有不同的数据表示形式，而同一数据也可能有不同的解释。

数据举例

- 数据：

（李明，男，1972，江苏，计算机系，1990）

- 李明是个大学生，男，1972年出生，江苏人，1990年考入计算机系
- 李明是位老师，男，1972年参加工作，江苏人，计算机系，1990年晋升为教授
- 数据的形式不能完全表达其内容，必须经过解释，**数据的解释**即对数据语义的说明



信息

二、数据库

- 数据库的定义
 - 数据库(**Database**, 简称**DB**)是长期储存在计算机内、有组织的、可共享的大量数据的集合。
- 数据库的基本特征
 - 数据按一定的数据模型组织、描述和储存
 - 可为各种用户共享
 - 冗余度较小
 - 数据独立性较高
 - 易扩展

三、数据库管理系统

- 什么是**DBMS**

- 位于用户与操作系统之间的一层数据管理软件。
- 是基础软件，是一个大型复杂的软件系统

- **DBMS**的用途

- 科学地组织和存储数据、高效地获取和维护数据

DBMS的主要功能

— 数据定义功能

提供数据定义语言(DDL)

定义数据库中的数据对象

— 数据组织、存储和管理

分类组织、存储和管理各种数据

确定组织数据的文件结构和存取方式

实现数据之间的联系

提供多种存取方法提高存取效率

DBMS的主要功能

– 数据操纵功能

提供数据操纵语言(DML)

实现对数据库的基本操作 (查询、插入、删除和修改)

– 数据库的事务管理和运行管理

数据库在建立、运行和维护时由DBMS统一管理和控制

保证数据的安全性、完整性、多用户对数据的并发使用

发生故障后的系统恢复

DBMS的主要功能

— 数据库的建立和维护功能(实用程序)

数据库初始数据装载转换

数据库转储

介质故障恢复

数据库的重组织

性能监视分析等

— 其它功能

DBMS与网络中其它软件系统的通信

两个DBMS系统的数据转换

异构数据库之间的互访和互操作

四、数据库系统

- 什么是数据库系统（**Database System**，简称**DBS**）

在计算机系统中引入数据库后的系统构成

- 数据库系统的构成
 - 数据库
 - 数据库管理系统（及其开发工具）
 - 应用系统
 - 数据库管理员

数据库系统中涉及的人员

- 数据库管理员
- 系统分析员和数据库设计人员
- 应用程序员
- 用户

1. 数据库管理员(DBA)

- 具体职责：
 - 决定数据库中的信息内容和结构
 - 决定数据库的存储结构和存取策略
 - 定义数据的安全性要求和完整性约束条件

— 监控数据库的使用和运行

- 周期性转储数据库

- 数据文件

- 日志文件

- 系统故障恢复

- 介质故障恢复

- 监视审计文件

— 数据库的改进和重组

§ 1.3 数据库系统的三级模式结构

§ 1.3.1 数据库模式的概念

§ 1.3.2 数据库系统内部的模式结构

从数据库管理系统角度看

1.3. 2 三级模式

- 外模式（External Schema）
- 模式（Schema）
- 内模式（Internal Schema）

1.模式（Schema）

- 模式（也称逻辑模式、数据库模式、概念模式）
 - 数据库中全体数据的逻辑结构的描述
 - 所有用户的公共数据视图，综合了所有用户的需求
- 一个数据库只有一个模式
- 模式的地位：是数据库系统模式结构的中间层
 - 与数据的物理存储细节和硬件环境无关
 - 与具体的应用程序、开发工具及高级程序设计语言无关

2. 外模式（External Schema）

- 外模式（也称子模式或用户模式）
 - 数据库用户（包括应用程序员和最终用户）使用的局部数据的逻辑结构的描述
 - 数据库用户的数据视图，是与某一应用有关的数据的逻辑表示

外模式（续）

- 外模式的地位：介于模式与应用之间
 - 模式与外模式的关系：一对多
 - 外模式通常是模式的子集
 - 一个数据库可以有多个外模式。反映了不同的用户的应用需求、看待数据的方式、对数据保密的要求
 - 对模式中同一数据，在外模式中的结构、类型、长度、保密级别等都可以不同
 - 外模式与应用的关系：一对多
 - 同一外模式也可以为某一用户的多个应用系统所使用，
 - 但一个应用程序只能使用一个外模式。

外模式（续）

- 外模式的用途

保证数据库安全性的一个有力措施。

每个用户只能看见和访问所对应的外模式中的数据

3. 内模式（Internal Schema）

- 内模式（也称存储模式）
 - 是数据物理结构和存储方式的描述
 - 是数据在数据库内部的表示方式
 - 记录的存储方式
 - 索引的组织方式
 - 数据是否压缩存储
 - 数据是否加密
- 一个数据库只有一个内模式

三级模式与二级映像

- 三级模式是对数据的三个抽象级别
- 二级映像在DBMS内部实现这三个抽象层次的联系和转换

1. 外模式 / 模式映像

- 定义外模式与模式之间的对应关系
- 每一个外模式都对应一个外模式 / 模式映像
- 映像定义通常包含在各自外模式的描述中

外模式 / 模式映象的用途

保证数据的逻辑独立性

- 当模式改变时，数据库管理员修改有关的外模式 / 模式映象，使外模式保持不变
- 应用程序是依据数据的外模式编写的，从而应用程序不必修改，保证了数据与程序的逻辑独立性，简称数据的逻辑独立性。

2. 模式 / 内模式映像

- 模式 / 内模式映像定义了数据全局逻辑结构与存储结构之间的对应关系。例如，说明逻辑记录和字段在内部是如何表示的
- 数据库中模式 / 内模式映像是唯一的
- 该映像定义通常包含在模式描述中

模式 / 内模式映象的用途

保证数据的物理独立性

- 当数据库的存储结构改变了（例如选用了另一种存储结构），数据库管理员修改模式 / 内模式映象，使模式保持不变
- 应用程序不受影响。保证了数据与程序的物理独立性，简称数据的物理独立性。

第2章 数据模型

2.1 概念模型

2.2 数据模型的组成要素

2.3 最常用的三种数据模型

数据模型(续)

- 客观对象的抽象过程---**两步抽象**
 - 现实世界中的客观对象抽象为概念模型;
概念模型 也称信息模型, 它是按用户的观点来对数据和信息建模。
 - 把概念模型转换为某一**DBMS**支持的数据模型。
数据模型 主要包括网状模型、层次模型、关系模型等, 它是按计算机系统的观点对数据建模。

1. 信息世界中的基本概念

(1) 实体 (Entity)

- 客观存在并可相互区别的事物称为实体。
- 可以是具体的人、事、物或抽象的概念。如一个学生，一本书，学生的一次选课

(2) 属性 (Attribute)

实体所具有的某一特性称为属性。

- 如：学生实体有学号、姓名、性别、出生日期、所在系别等方面的属性。
- 属性有“**型**”和“**值**”之分，“**型**”即为属性名，如姓名、年龄、性别是属性的型；“**值**”即为属性的具体内容，如
(990001, 张立, 20, 男, 计算机) 这些属性值的集合表示了一个学生实体。

信息世界中的基本概念(续)

(3) 码 (Key)

能唯一标识实体的属性或属性集称为码。

— 如学生的学号。

(4) 域 (Domain)

属性的取值范围称为该属性的域。

— 如学号的域为9位整数，姓名的域为字符串集合，年龄的域为小于40的整数，性别的域为（男，女）。

信息世界中的基本概念(续)

(5) 实体型 (Entity Type)

用实体名及其属性名集合来抽象和刻画同类实体称为实体型。

- 如学生 (学号, 姓名, 年龄, 性别, 系) 就是一个实体型。

(6) 实体集 (Entity Set)

同型实体的集合称为实体集。

- 如所有的学生、所有的课程等。

(7) 联系 (Relationship)

现实世界中事物内部以及事物之间的联系在信息世界中反映为实体型内部的联系和实体型之间的联系

联系

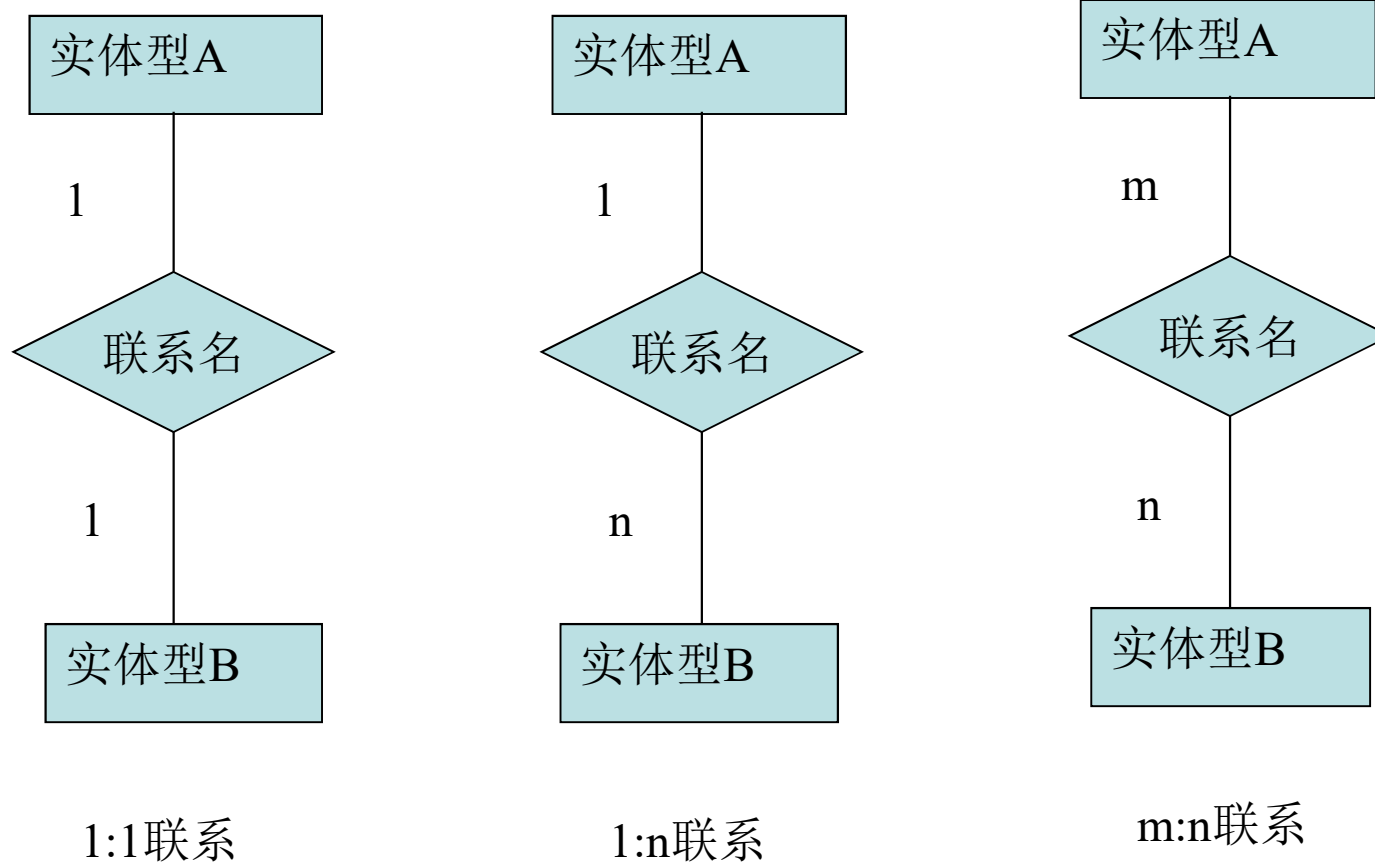
两个实体型间

三个或三个以上实体型间

同一个实体型内

$\left[\begin{array}{l} \text{一对一联系 (1:1)} \\ \text{一对多联系 (1:n)} \\ \text{多对多联系 (m:n)} \end{array} \right]$

两个实体型间的联系



两个实体型间的联系

- 一对一联系

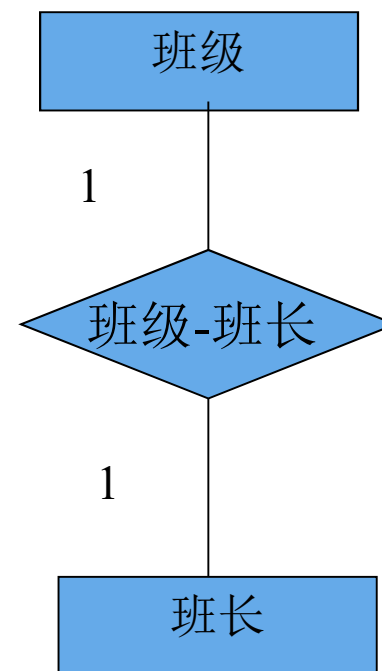
- 如果对于实体集A中的每一个实体，实体集B中至多有一个实体与之联系，反之亦然，则称实体集A与实体集B具有一对一联系。记为1:1。

- 实例

班级与班长之间的联系：

一个班级只有一个正班长

一个班长只在一个班中任职



1:1联系

两个实体型间的联系 (续)

- 多对多联系 ($m:n$)
 - 如果对于实体集A中的每一个实体，实体集B中有 n 个实体 ($n>1$) 与之联系，反之，对于实体集B中的每一个实体，实体集A中也有 m 个实体 ($m>1$) 与之联系，则称实体集A与实体B具有多对多联系。记为 $m:n$ 。
 - 实例
 - 课程与学生之间的联系：
 - 一门课程同时有若干个学生选修
 - 一个学生可以同时选修多门课程

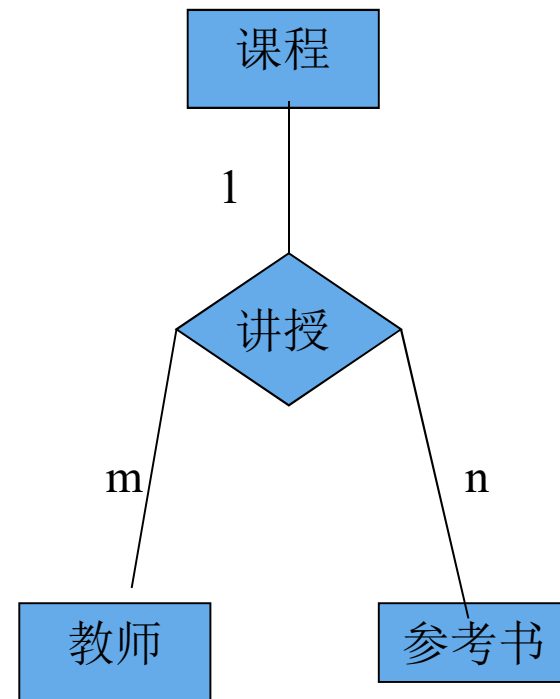
多个实体型间的联系

- 两个以上不同实体型之间也可能存在各种联系，以3个不同实体型A、B、C为例，它们之间的典型联系有1:m:n和m:n:p联系。
- 对于1:m:n联系，表示A和B之间是1:m(一对多)联系，B和C之间m:n(多对多)联系，A和C之间是1:n(一对多)联系。

多个实体型间的联系(续)

- 实例

- 课程、教师与参考书三个实体型
- 如果一门课程可以有若干个教师讲授, 使用若干本参考书, 每一个教师只讲授一门课程, 每一本参考书只供一门课程使用, 课程与教师、参考书之间的联系是一对多的



两个以上实体型间1:n联系

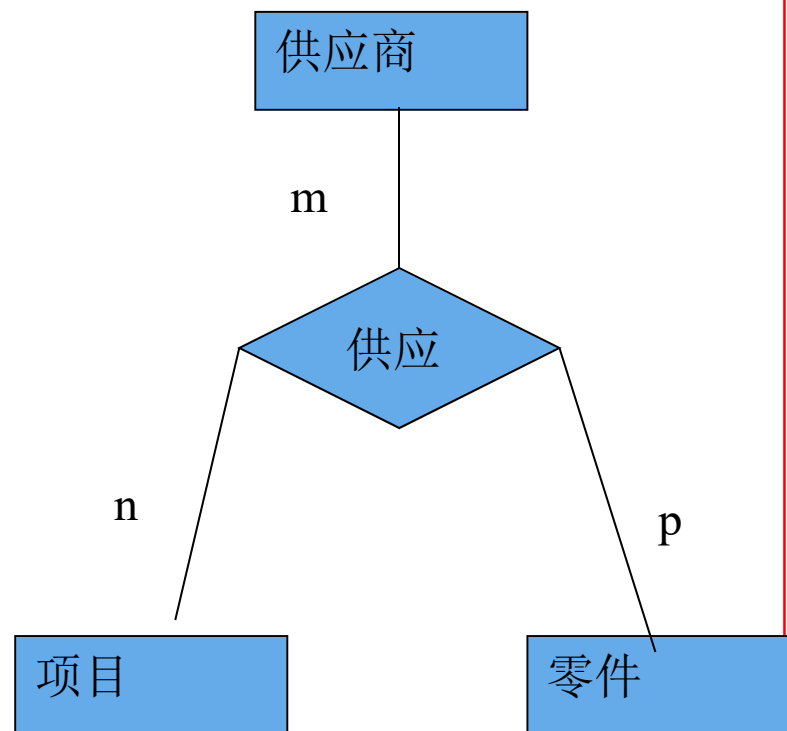
两个以上实体型之间的联系(续)

- ❖ 多个实体型间的一对一联系
- ❖ 两个以上实体型间的多对多联系

■ 实例

供应商、项目、零件三个实体型:

一个供应商可以供给多个项目多种零件 ;每个项目可以使用多个供应商供应的零件 ;每种零件可由不同供应商供给



两个以上实体型间m:n联系

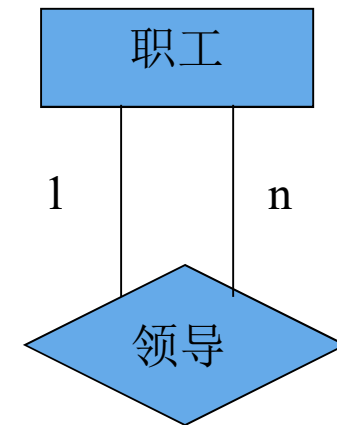
同一实体型内部的联系

- 实例

职工实体型内部具有领导与被领导的联系

某一职工（干部）“领导”若干名职工；一个职工仅被另外一个职工直接领导

这是一对多的联系



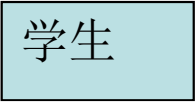
单个实体型内部1:n
联系

2、 概念模型的表示方法

- 实体—联系方法(E-R方法)
 - 用E-R图来描述现实世界的概念模型

E-R图

- 实体型
 - 用矩形表示，矩形框内写明实体型名。



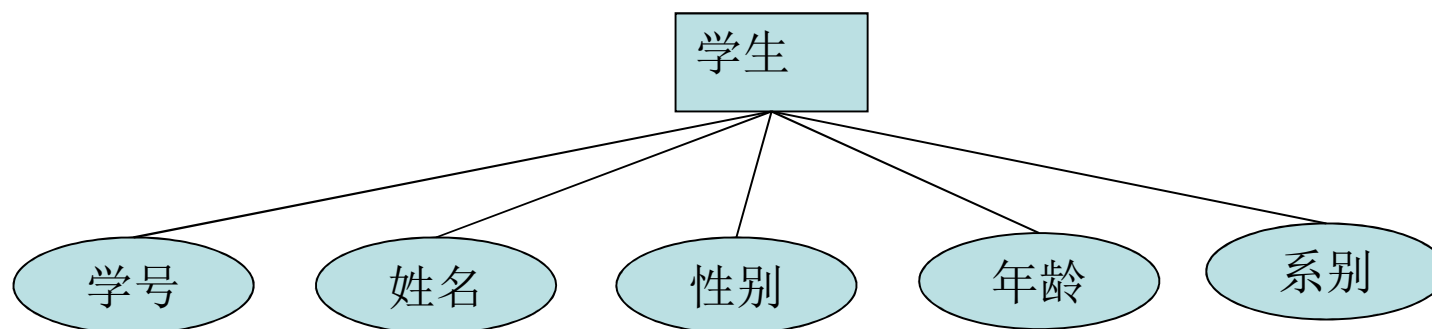
学生



教师

E-R图(续)

- 属性
 - 用椭圆形表示，并用无向边将其与相应的实体型连接起来。

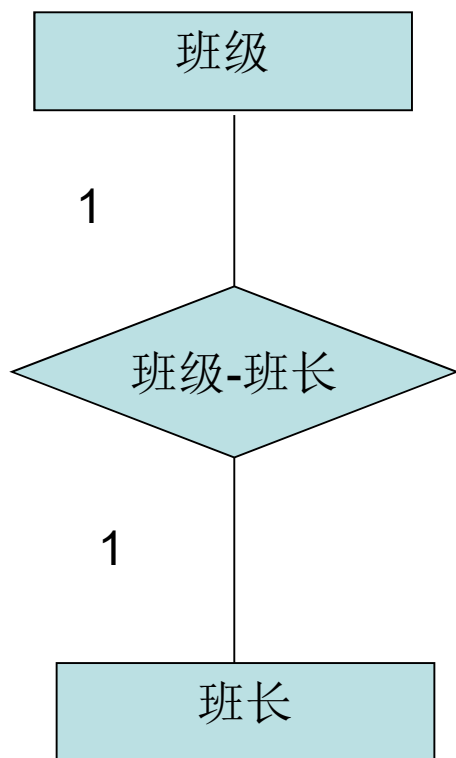


E-R图(续)

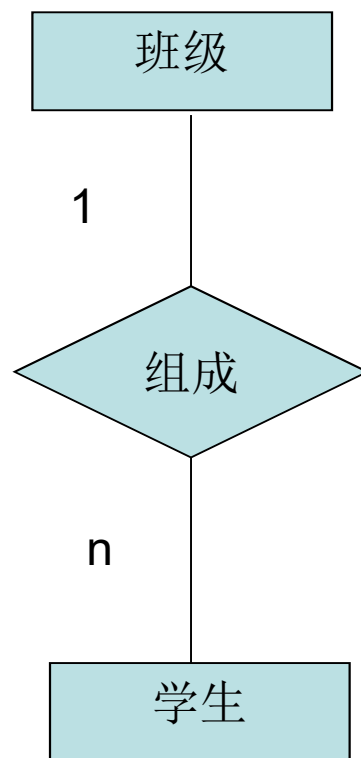
- 联系

- 联系本身：用菱形表示，菱形框内写明联系名，并用无向边分别与有关实体型连接起来，同时无向边旁标上联系类型（1:1、1:n或m:n）
- 联系的属性：联系本身也是一种实体型，也可以有属性。如果一个联系具有属性，则这些属性也要用无向边与该联系连接起来。

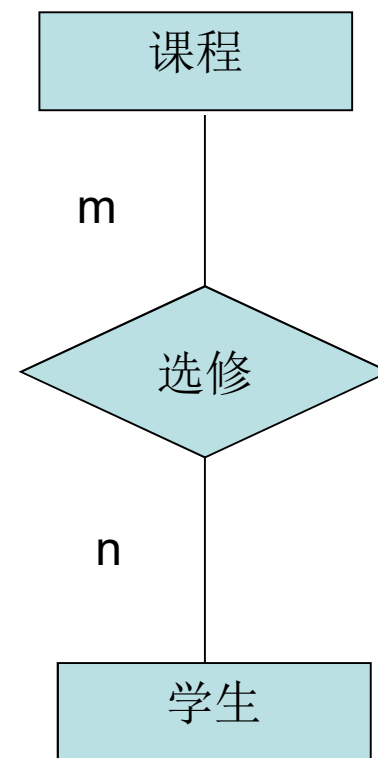
联系的表示方法示例



1:1联系

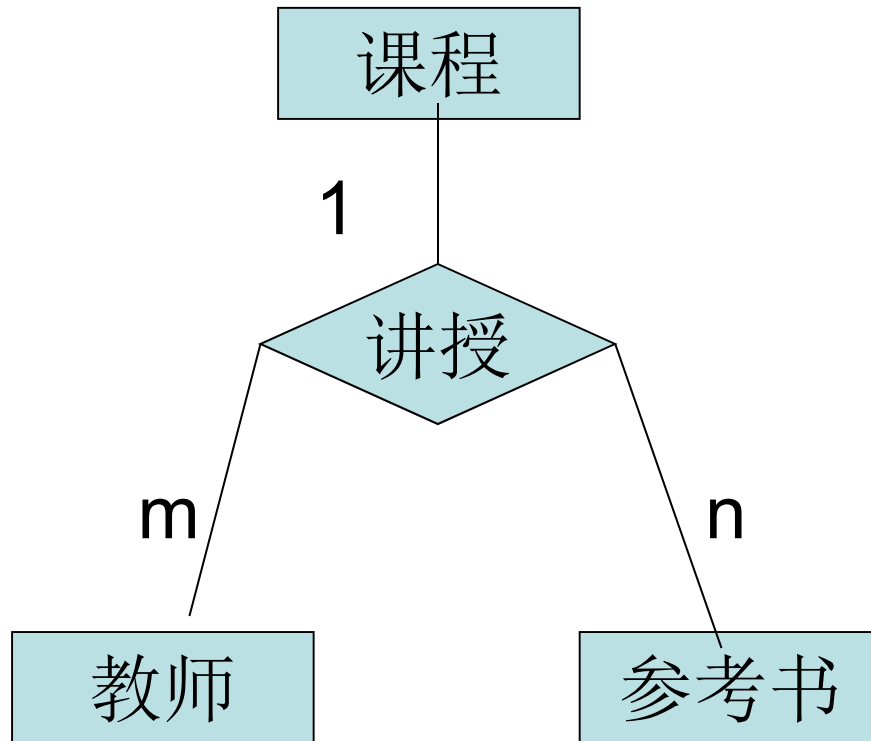


1:n联系

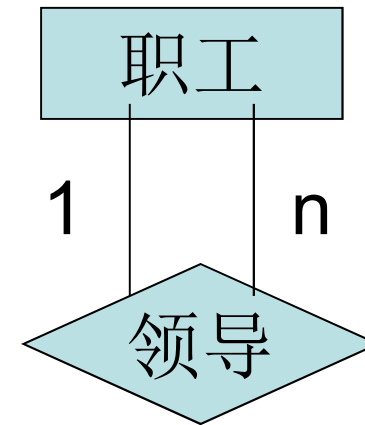


m:n联系

联系的表示方法示例（续）

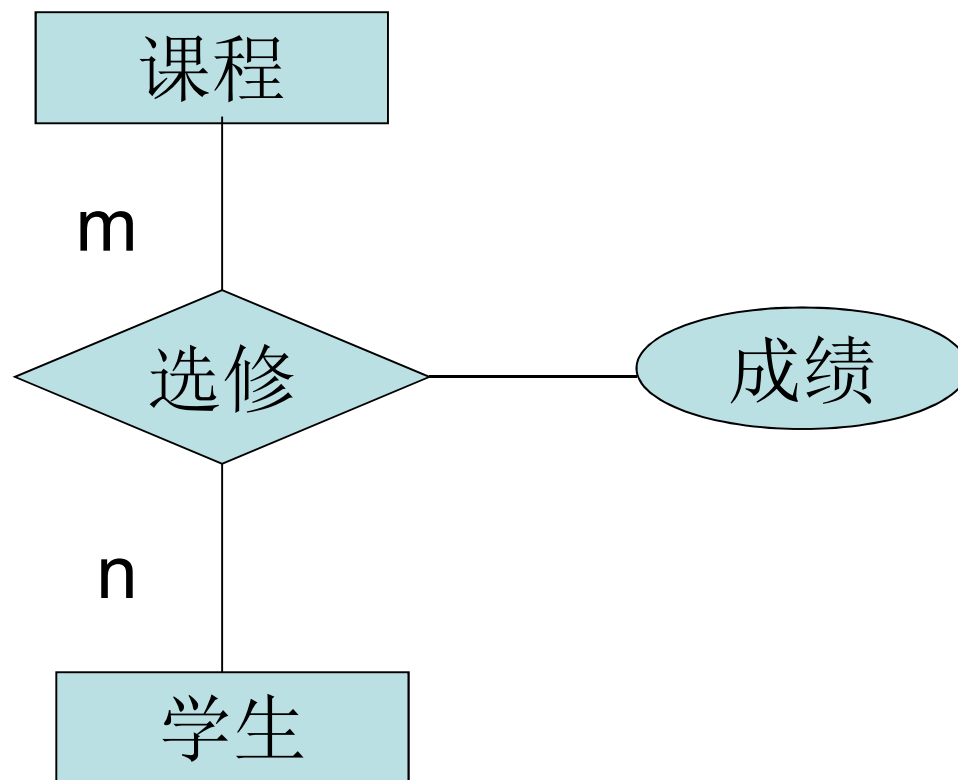


多个实体型间的1:n联系



同一实体型内部的1:n联系

联系属性的表示方法



如何建立实体-联系模型

- (1) 了解用户需要用数据库解决那些问题
- (2) 确定实体模型应包含那些实体才能满足用户需要解决的问题；
- (3) 这些实体中哪些实体是明显的；
- (4) 根据考虑问题的范围和角度，确定是否有实体间的某种联系或某个实体的某个属性也应视为概念存在的实体；
- (5) 根据现有技术条件，实际能够观测和存储那些实体和那些属性，等等。

例1.2.3：试设计一个实体-联系模型以存放初二年级期末考试语文、数学、英语、物理、化学五门课程学生成绩。

E-R图实例：某工厂物资管理概念模型

用E-R图表示某个工厂物资管理的概念模型

- 实体

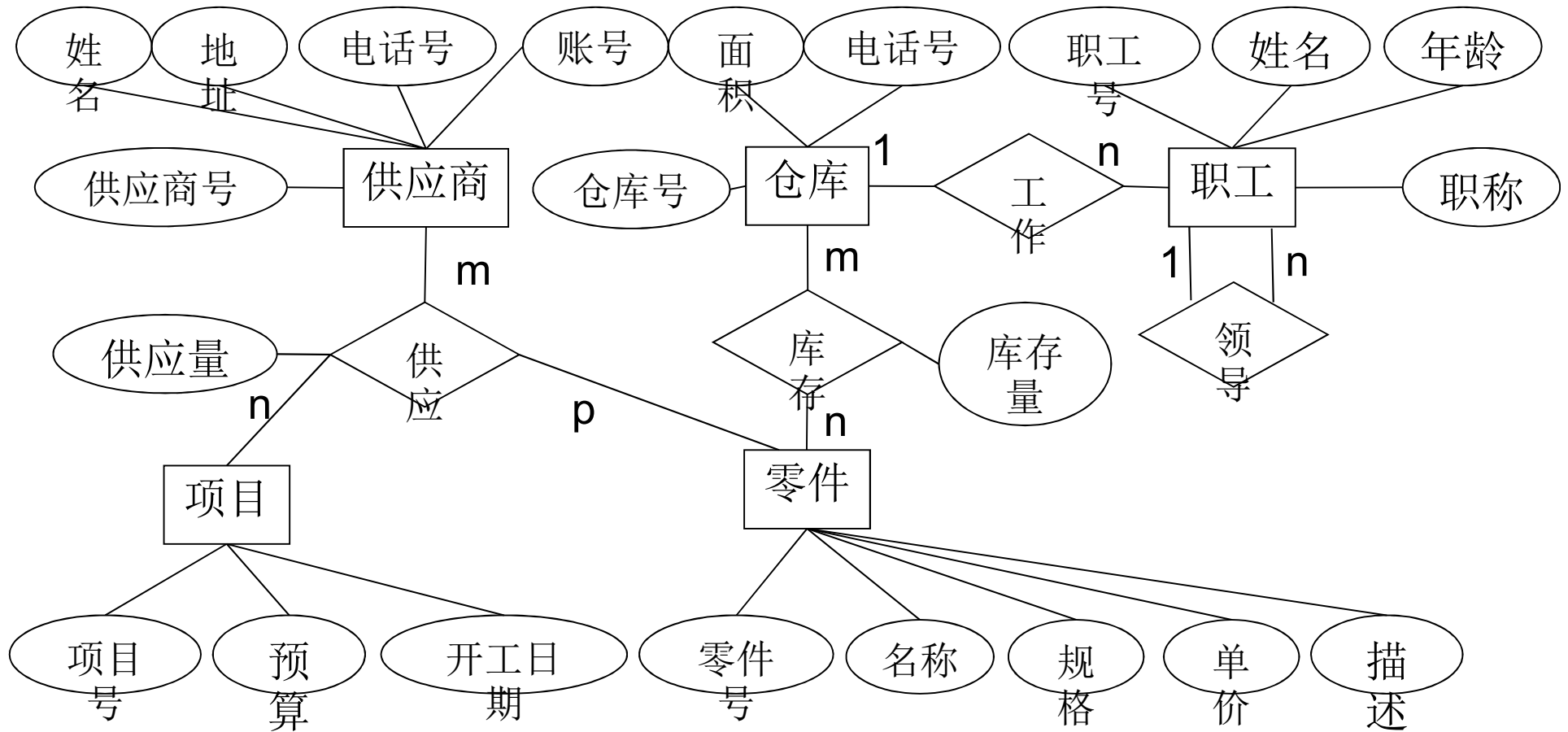
- 仓库： 仓库号、面积、电话号码
- 零件： 零件号、名称、规格、单价、描述
- 供应商： 供应商号、姓名、地址、电话号码、帐号
- 项目： 项目号、预算、开工日期
- 职工： 职工号、姓名、年龄、职称

E-R图实例：某工厂物资管理概念模型

- 实体之间的联系如下：

- (1)一个仓库可以存放多种零件，一种零件可以存放在多个仓库中。
用库存量来表示某种零件在某个仓库中的数量。
- (2)一个仓库有多个职工当仓库保管员，一个职工只能在一个仓库工作。职工实体型中具有一对多的联系
- (3)职工之间具有领导-被领导关系。即仓库主任领导若干保管员。
- (4)供应商、项目和零件三者之间具有多对多的联系

E-R图实例：某工厂物资管理概念模型



在简单的教务管理系统中，包括四个实体，分别为：

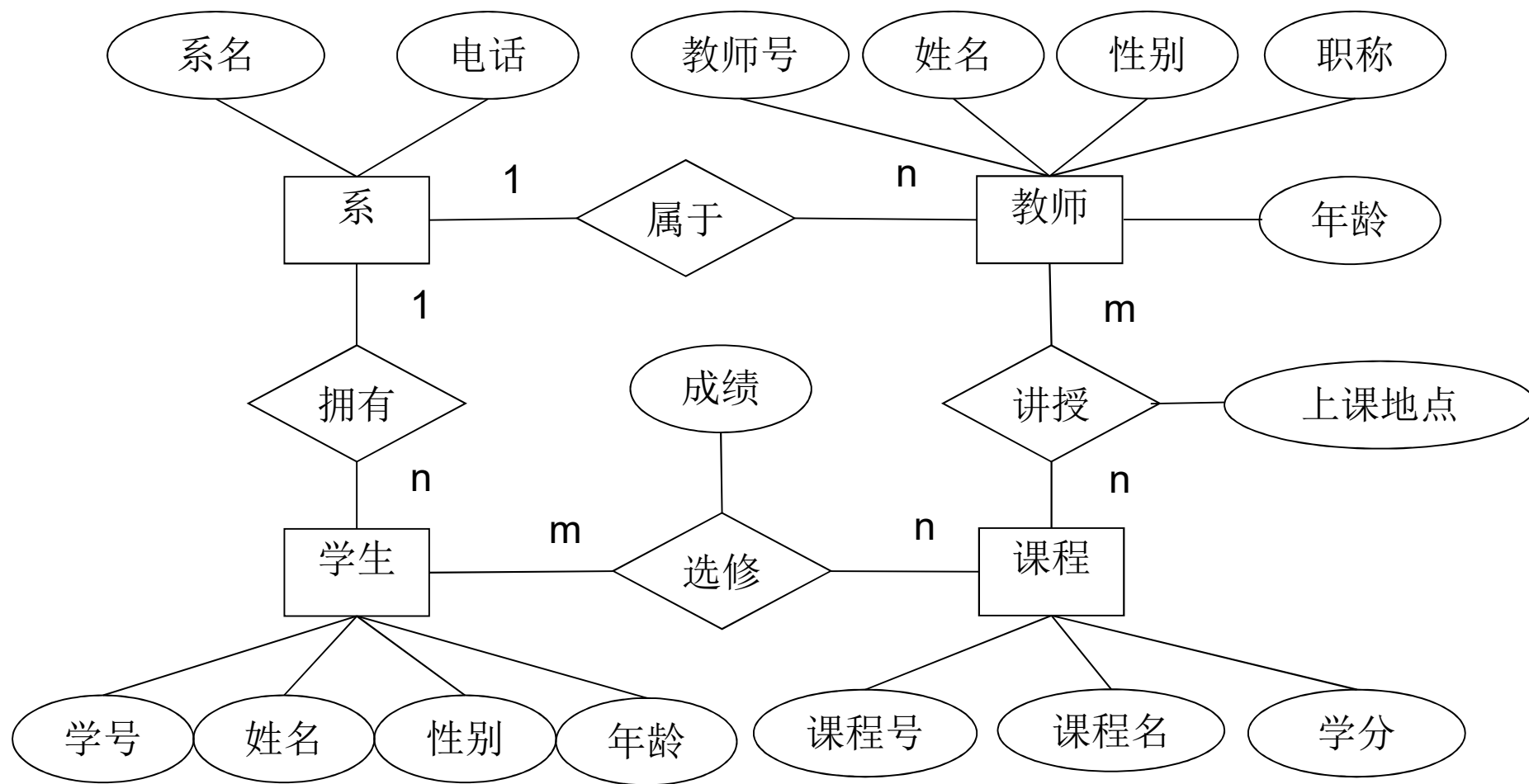
- 系：系名，电话
- 教师：教师号，姓名，性别，职称，年龄
- 学生：学号，姓名，性别，年龄
- 课程：课程号，课程名，学分

且存在如下语义约束：

- 一个系可拥有多个教师，一个教师只能属于一个系。
- 一个系可拥有多个学生，一个学生只能属于一个系。
- 一个学生可选修多门课程，一门课程可为多个学生选修，
每一个学生选修每门课程都有一个成绩。
- 一个教师可讲授多门课程，一门课程可为多个教师讲授。

画出**E-R**图。

简单教务管理系统的E-R图



思考

- 上面两个例子，转换成关系模式，如何进行？
- 把“数据库设计”部分的“逻辑结构设计”放到这一部分联合复习

7.4 逻辑结构设计

7.4.1 E-R图向关系模型的转换

7.4.2 数据模型的优化

7.4.3 设计用户子模式

7.4.1 E-R图向关系模型的转换

- 转换内容
- 转换原则

E-R图向关系模型的转换（续）

- E-R图向关系模型的转换要解决的问题
 - 如何将实体型和实体间的联系转换为关系模式
 - 如何确定这些关系模式的属性和码
- 转换内容
 - 将E-R图转换为关系模型：将实体、实体的属性和实体之间的联系转换为关系模式。

E-R图向关系模型的转换（续）

实体型间的联系有以下不同情况：

(1) 一个1:1联系可以转换为一个独立的关系模式，也可以与任意一端对应的关系模式合并。

- 转换为一个独立的关系模式
- 与某一端实体对应的关系模式合并

(2) 一个1:n联系可以转换为一个独立的关系模式，也可以与n端对应的关系模式合并。

- 转换为一个独立的关系模式
- 与n端对应的关系模式合并

E-R图向关系模型的转换（续）

(3) 一个m:n联系转换为一个关系模式。

例，“选修”联系是一个m:n联系，可以将它转换为如下关系模式，其中学号与课程号为关系的组合码：

选修（学号，课程号，成绩）

E-R图向关系模型的转换（续）

(4)三个或三个以上实体间的一个多元联系转换为一个关系模式。

例，“讲授”联系是一个三元联系，可以将它转换为如下关系模式，其中课程号、职工号和书号为关系的组合码：

讲授（课程号，职工号，书号）

E-R图向关系模型的转换（续）

(5)具有相同码的关系模式可合并

- 目的：减少系统中的关系个数
- 合并方法：将其中一个关系模式的全部属性加入到另一个关系模式中，然后去掉其中的同义属性（可能同名也可能不同名），并适当调整属性的次序

思考

- 上面两个例子，转换成关系模式

§ 2.3 常用数据模型

- 非关系模型
 - 层次模型(Hierarchical Model)
 - 网状模型(Network Model)
- 关系模型(Relational Model)
- 面向对象模型(Object Oriented Model)

一、 层次模型

- 层次模型是数据库系统中最早出现的数据模型，采用层次模型的数据库的典型代表是IBM公司的IMS（Information Management System）数据库管理系统。
- 现实世界中，许多实体之间的联系都表现出一种很自然的层次关系，如家族关系，行政机构等。

1. 层次数据模型的数据结构

- 层次模型的数据结构：“有向树”

实体型：用记录类型描述。

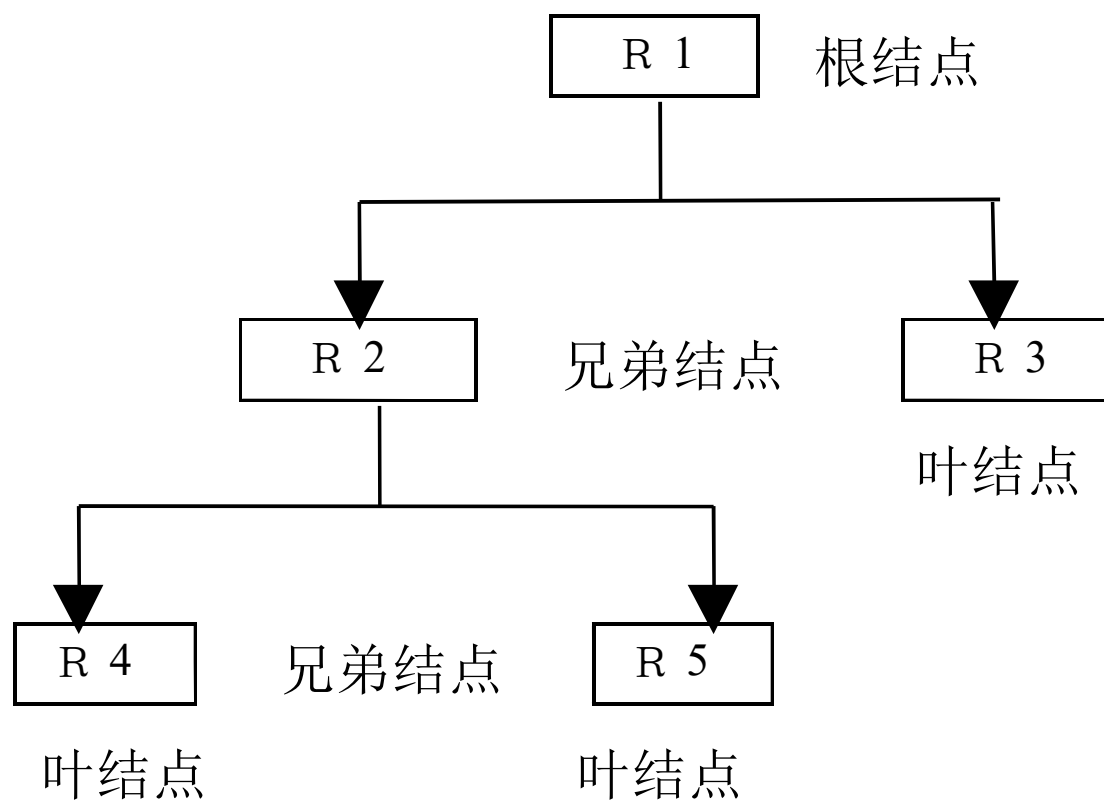
每个结点表示一个记录类型。

属性：用字段描述。每个记录类型可包含若干个字段。

联系：用结点之间的连线（有向边）表示记录类型之间的一对多的联系

- 层次模型中的几个术语
 - 根结点，双亲结点，兄弟结点，叶结点

层次数据模型的数据结构（续）



层次数据模型的数据结构（续）

层次模型的特征：

- （1）有且仅有一个结点没有双亲，该结点就是根结点；
- （2）根结点以外的其他结点有且仅有一个双亲结点，这就使得层次数据库系统只能直接处理一对多的实体关系；

二、网状模型

- 20世纪70年代，数据系统语言研究会CODASYL（Conference On Data System Language）下属的数据库任务组DBTG（Data Base Task Group）提出了一个系统方案，**DBTG系统**，也称CODASYL系统，成为了网状数据模型的代表。
- 实际系统
 - Cullinet Software 公司的 IDMS
 - Univac公司的 DMS1100
 - Honeywell公司的IDS/2
 - HP公司的IMAGE

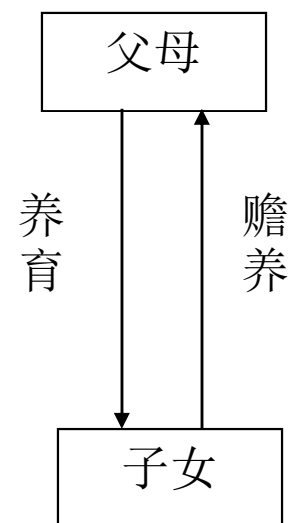
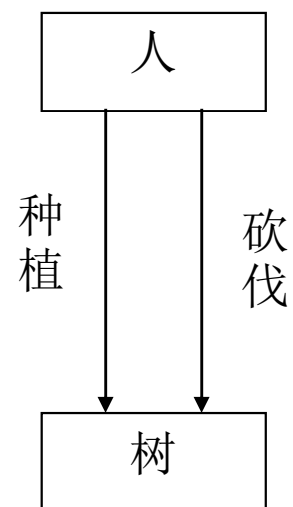
1.网状数据模型的数据结构

- 网状模型

满足下面两个条件的基本层次联系的集合为网状模型。

1. 允许一个以上的结点无双亲；
2. 一个结点可以有多个的双亲。

网状数据模型的数据结构（续）



三、关系模型

- 1970年美国IBM公司的研究员E.F.Codd首次提出了数据库系统的关系模型。
 - 他发表了题为“大型共享银行数据的关系模型”（A Relation Model of Data for Large Shared Data Banks）,在文中解释了关系模型，定义了某些关系代数运算，研究了数据的函数相关性，定义了关系的第三范式，从而开创了数据库的关系方法和数据规范化理论的研究，为关系数据库技术奠定了理论基础。
 - 他因此获得了1981年的图灵奖。

关系模型的基本概念

- 关系（Relation）

一个关系对应通常说的一张表。

- 元组（Tuple）

表中的一行即为一个元组。

- 属性（Attribute）

表中的一列即为一个属性，给每一个属性起一个名称即属性名。

关系模型的基本概念

- 分量

元组中的一个属性值。

- 域 (Domain)

属性的取值范围。

- 关系模式

对关系的描述

关系名 (属性1, 属性2, ..., 属性n)

学生 (学号, 姓名, 年龄, 性别, 系, 年级)

2. 定义关系模式

关系模式可以形式化地表示为：

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

R 关系名

U 组成该关系的属性名集合

D 属性组 U 中属性所来自的域

DOM 属性向域的映象集合

F 属性间的数据依赖关系集合

定义关系模式 (续)

关系模式通常可以简记为

$R(U)$ 或 $R(A_1, A_2, \dots, A_n)$

■ R : 关系名

■ A_1, A_2, \dots, A_n : 属性名

注：域名及属性向域的映象常常直接说明为
属性的类型、长度

关系模型的基本概念

6) 码

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

简单的情况：候选码只包含一个属性

全码 (All-key)

最极端的情况：关系模式的所有属性组是这个关系模式的候选码，称为全码 (All-key)

关系模型的基本概念

码(续)

主码

若一个关系有多个候选码，则选定其中一个为主码（Primary key）

主属性

候选码的诸属性称为主属性（Prime attribute）

不包含在任何侯选码中的属性称为非主属性（ Non-Prime attribute）

或非码属性（Non-key attribute）

关系数据模型的数据结构(续)

- 实体及实体间的联系的表示方法
 - 实体型：直接用关系（表）表示。
 - 属性：用属性名表示。
 - 联系：用键表示。
- 在层次和网状模型中，联系用指针来实现，在关系模型中，记录之间的联系通过键来实现
- 关系模型的操作语言是SQL

关系数据模型的数据结构（续）

- ❖ 关系必须是规范化的，满足一定的规范条件
最基本的规范条件：关系的每一个分量必须是一个不可分的数据项，
不允许表中还有表
图1.27中工资和扣除是可分的数据项，不符合关系模型要求

职工号	姓名	职 称	工 资			扣 除		实 发
			基 本	津 贴	职 务	房 租	水 电	
86051	陈 平	讲 师	1305	1200	50	160	112	2283

图1.27 一个工资表(表中有表)实例

关系的完整性

关系的三类完整性约束

- 实体完整性
- 参照完整性
- 用户定义的完整性

关系的三类完整性约束

- 实体完整性和参照完整性：

关系模型必须满足的完整性约束条件

称为关系的两个不变性，应该由关系系统自动支持

- 用户定义的完整性：

应用领域需要遵循的约束条件，体现了具体领域中的语义约束

实体完整性

规则2.1 实体完整性规则 (**Entity Integrity**)

若属性 A 是基本关系 R 的主属性, 则属性 A 不能取空值

例:

SAP(SUPERVISOR, SPECIALITY, POSTGRADUATE)

POSTGRADUATE:

主码 (假设研究生不会重名)

不能取空值

实体完整性(续)

实体完整性规则的说明

- (1) 实体完整性规则是针对基本关系而言的。一个基本表通常对应现实世界的一个实体集。
- (2) 现实世界中的实体是可区分的，即它们具有某种唯一性标识。
- (3) 关系模型中以主码作为唯一性标识。
- (4) 主码中的属性即主属性不能取空值。

主属性取空值，就说明存在某个不可标识的实体，即存在不可区分的实体，这与第（2）点相矛盾，因此这个规则称为实体完整性

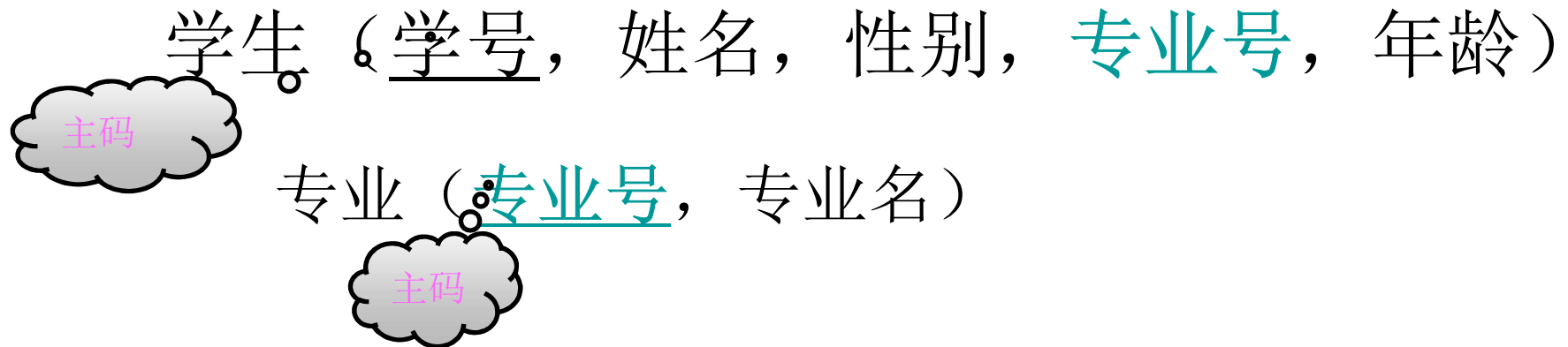
参照完整性

1. 关系间的引用
2. 外码
3. 参照完整性规则

1. 关系间的引用

- 在关系模型中实体及实体间的联系都是用关系来描述的，因此可能存在着关系与关系间的引用。

例1 学生实体、专业实体



- ❖ 学生关系引用了专业关系的主码“专业号”。
- ❖ 学生关系中的“专业号”值必须是确实存在的专业的专业号，即专业关系中有该专业的记录。

关系间的引用(续)

例2 学生、课程、学生与课程之间的多对多
联系

学生（学号，姓名，性别，专业号，年龄）

课程（课程号，课程名，学分）

选修（学号，课程号，成绩）

关系间的引用(续)

例3 学生实体及其内部的一对多联系

学生（学号，姓名，性别，专业号，年龄，班长）

学号	姓名	性别	专业号	年龄	班长
801	张三	女	01	19	802
802	李四	男	01	20	
803	王五	男	01	20	802
804	赵六	女	02	20	805
805	钱七	男	02	19	

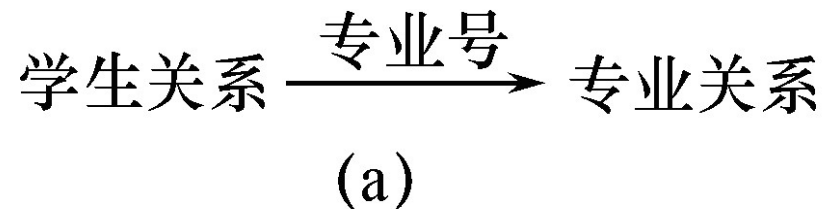
- ❖ “学号”是主码，“班长”是外码，它引用了本关系的“学号”
- ❖ “班长”必须是确实存在的学生的学号

2. 外码 (Foreign Key)

- 设 F 是基本关系 R 的一个或一组属性，但不是关系 R 的码。如果 F 与基本关系 S 的主码 K_s 相对应，则称 F 是基本关系 R 的**外码**
- 基本关系 R 称为**参照关系** (Referencing Relation)
- 基本关系 S 称为**被参照关系** (Referenced Relation) 或**目标关系** (Target Relation)

外码(续)

- [例1]：学生关系的“专业号”与专业关系的主码“专业号”相对应
 - “专业号”属性是学生关系的外码
 - 专业关系是被参照关系，学生关系为参照关系



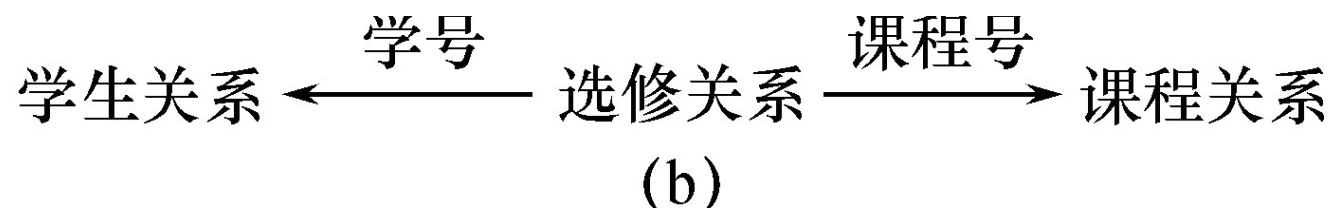
外码(续)

- [例2] :

选修关系的“学号”与学生关系的主码“学号”相对应

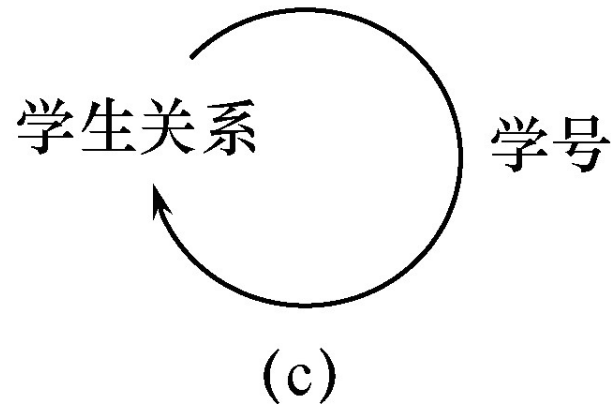
选修关系的“课程号”与课程关系的主码“课程号”相对应

- “学号”和“课程号”是选修关系的外码
- 学生关系和课程关系均为被参照关系
- 选修关系为参照关系



外码(续)

- [例3]：“班长”与本身的主码“学号”相对应
 - “班长”是外码
 - 学生关系既是参照关系也是被参照关系



外码(续)

- 关系 R 和 S 不一定是不同的关系
- 目标关系 S 的主码 K_s 和参照关系的外码 F 必须定义在同一个（或一组）域上
- 外码并不一定要与相应的主码同名

当外码与相应的主码属于不同关系时，往往取相同的名字，以便于识别

参照完整性规则

规则2.2 参照完整性规则

若属性（或属性组） F 是基本关系 R 的外码它与基本关系 S 的主码 K_s 相对应（基本关系 R 和 S 不一定是不同的关系），则对于 R 中每个元组在 F 上的值必须为：

- 或者取空值（ F 的每个属性值均为空值）
- 或者等于 S 中某个元组的主码值

参照完整性规则(续)

[例1]:

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

参照完整性规则(续)

〔例2〕：

选修（学号，课程号，成绩）

“学号”和“课程号”可能的取值：

（1）选修关系中的主属性，不能取空值

（2）只能取相应被参照关系中已经存在的主码值

参照完整性规则(续)

例3)：

学生 (学号，姓名，性别，专业号，年龄，班长)

“班长”属性值可以取两类值：

- (1) 空值，表示该学生所在班级尚未选出班长
- (2) 非空值，该值必须是本关系中某个元组的学号值

用户定义的完整性

- 针对某一具体关系数据库的约束条件，反映某一具体应用所涉及的数据必须满足的语义要求
- 关系模型应提供定义和检验这类完整性的机制，以便使用统一的系统的方法处理它们，而不要由应用程序承担这一功能

用户定义的完整性(续)

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}

关系数据库模型的操作

- ❖ 数据操作是集合操作，操作对象和操作结果都是关系
 - 查询 / 插入 / 删除 / 更新
- ❖ 数据操作是集合操作，操作对象和操作结果都是关系，即若干元组的集合
- ❖ 存取路径对用户隐蔽，用户只要指出“干什么”，不必详细说明“怎么干”

基本关系操作

- 关系操作的特点

- 集合操作方式：操作的对象和结果都是集合，一次一集合的方式

- 常用的关系操作

- 查询：选择、投影、连接、除、并、交、差
- 数据更新：插入、删除、修改
- 查询的表达能力是其中最主要的部分
- 选择、投影、并、差、笛卡尔基是5种基本操作

关系操作

- 查询是最主要的部分。所以说，关系数据库的核心部分是查询，故又称为查询语言，而查询的条件要使用关系运算表达式来表示
- 按表达查询的方法不同，关系运算可分为关系代数和关系演算两大类。

关系代数

- 关系代数是对关系进行集合代数运算，是基于关系代数的操作语言，称为关系代数语言，简称关系代数。
 - 它是由IBM在一个实验性的系统上实现的，称为ISBL(Information System Base Language)语言。
 - ISBL的每个语句都类似于一个关系代数表达式。
- 关系代数的运算对象是关系，运算结果也是关系

§ 3.2 关系代数

关系代数的运算按运算符的不同主要分为两类：

- **传统的集合运算**：把关系看成**元组**的集合，以元组作为集合中元素来进行运算，其运算是从关系的“**水平**”方向即行的角度进行的。包括并、差、交和笛卡尔积等运算。
- **专门的关系运算**：不仅涉及行运算，也涉及列运算，这种运算是为数据库的应用而引进的特殊运算。包括选取、投影、连接和除法等运算。

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	笛卡尔积		\leq	小于等于
				$=$	等于
				\neq	不等于

关系代数运算符（续）

运算符	含义		运算符	含义	
专门的关系运算符	σ π \bowtie \div	选择 投影 连接 除	逻辑运算符	\neg \wedge \vee	非 与 或

传统的集合运算

对两个关系的集合运算传统的集合运算是二目运算，是在两个关系中进行的。但是并不是任意的两个关系都能进行这种集合运算，而是要在两个满足一定条件的关系中进行运算。

- 相容性定义：设给定两个关系 R 、 S ，若满足：
 - (1) 具有相同的度 n ;
 - (2) R 中第 i 个属性和 S 中第 i 个属性必须来自同一个域。

则说关系 R 、 S 是相容的。除笛卡尔积外，要求参加运算的关系必须满足上述的相容性定义。

【例】 如图 (a)、(b)所示的两个关系***R***与***S***为相容关系

R

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

(a)

S

A	B	C
a1	b1	c1
a2	b2	c1
a2	b3	c2

(b)

R

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

(a)

S

A	B	C
a1	b1	c1
a2	b2	c1
a2	b3	c2

(b)

RUS

A	B	C
a1	B1	c1
a1	B1	c2
a2	B2	c1
a2	B3	c2

Select * from R Union Select * from S
注意，不是union all

R

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

(a)

S

A	B	C
a1	b1	c1
a2	b2	c1
a2	b3	c2

(b)

R-S

A	B	C
a1	b1	c2

(1)select * from R except select * from S

(2)select * from R where not exists

(select * from S where R.A=S.A and R.B=S.B and R.C=S.C)

R

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

(a)

S

A	B	C
a1	b1	c1
a2	b2	c1
a2	b3	c2

(b)

R ∩ S

A	B	C
a1	B1	c1
a2	B2	c1

(1) select * from R intersect select * from S

(2) Select R.* from R ,S where

R.A=S.A and R.B=S.B and R.C=S.C

传统的集合运算

4. 广义笛卡尔积 (Extended Cartesian Product)

- 两个分别为n目和m目关系R和S的广义笛卡尔积是一个(n+m)列的元组的集合，元组的前n列是关系R的一个元组，后m列是关系S的一个元组。若R有 k_1 个元组，S有 k_2 个元组，则关系R和关系S的广义笛卡尔积有 k_1*k_2 个元组，记作

$$R \times S = \{t_r \cup t_s \mid t_r \in R, \wedge t_s \in S\}$$

- 关系的广义笛卡尔积可用于两关系的连接操作

R×S

R.A	R.B	R.C	S.A	S.B	S.C
a1	b1	c1	a1	b1	c1
a1	b1	c1	a2	b2	c1
a1	b1	c1	a2	b3	c2
a1	b1	c2	a1	b1	c1
a1	b1	c2	a2	b2	c1
a1	b1	c2	a2	b3	c2
a2	b2	c1	a1	b1	c1
a2	b2	c1	a2	b2	c1
a2	b2	c1	a2	b3	c2

Select * from R ,S

专门的关系运算(续)

- 选择
- 投影
- 连接
- 除

专门的关系运算(续)

4) 学生-课程数据库:

学生关系Student、课程关系Course和选修关系SC

Student

学号 Sno	姓名 Sname	性别 Ssex	年龄 Sage	所在系 Sdept
200215121	李勇	男	20	CS
200215122	刘晨	女	19	IS
200215123	王敏	女	18	MA
200215125	张立	男	19	IS

(a)

专门的关系运算(续)

Course

课程号 Cno	课程名 Cname	先行课 Cpno	学分 Ccredit
1	数据库	5	4
2	数学		2
3	信息系统	1	4
4	操作系统	6	3
5	数据结构	7	4
6	数据处理		2
7	PASCAL 语言	6	4

(b)

An Introduction to Database System

专门的关系运算(续)

SC

学号 Sno	课程号 Cno	成绩 Grade
200215121	1	92
200215121	2	85
200215121	3	88
200215122	2	90
200215122	3	80

(c)

选择（续）

[例1] 查询信息系（IS系）全体学生

$\sigma_{Sdept = 'IS'} (Student)$
或 $\sigma_5 = 'IS' (Student)$

结果：

Sno	Sname	Ssex	Sage	Sdept
200215122	刘晨	女	19	IS
200215125	张立	男	19	IS

选择（续）

[例2] 查询年龄小于20岁的学生

$\sigma_{\text{Sage} < 20}(\text{Student})$

或 $\sigma_4 < 20(\text{Student})$

结果：

Sno	Sname	Ssex	Sage	Sdept
200215122	刘晨	女	19	IS
200215123	王敏	女	18	MA
200215125	张立	男	19	IS

投影（续）

- [例3] 查询学生的姓名和所在系
即求**Student**关系上学生姓名和所在系两个属性上的投影

$\pi_{\text{Sname, Sdept}}(\text{Student})$
或 $\pi_{2, 5}(\text{Student})$

结果：

投影（续）

Sname	Sdept
李勇	CS
刘晨	IS
王敏	MA
张立	IS

投影（续）

[例4] 查询学生关系Student中都有哪些系

$\pi_{\text{Sdept}}(\text{Student})$

结果:

Sdept
CS
IS
MA

连接(续)

- [例5]关系 R 和关系 S 如下所示:

R		
A	B	C
a_1	b_1	5
a_1	b_2	6
a_2	b_3	8
a_2	b_4	12

S	
B	E
b_1	3
b_2	7
b_3	10
b_3	2
b_5	2

连接(续)

一般连接 $R \bowtie_{C < E} S$ 的结果如下:

$R \bowtie_{C < E} S$				
A	$R.B$	C	$S.B$	E
a_1	b_1	5	b_2	7
a_1	b_1	5	b_3	10
a_1	b_2	6	b_2	7
a_1	b_2	6	b_3	10
a_2	b_3	8	b_3	10

连接(续)

等值连接 $R \bowtie_{R.B=S.B} S$ 的结果如下:

A	$R.B$	C	$S.B$	E
a_1	b_1	5	b_1	3
a_1	b_2	6	b_2	7
a_2	b_3	8	b_3	10
a_2	b_3	8	b_3	2

连接(续)

自然连接 $R \bowtie S$ 的结果如下:

A	B	C	E
a_1	b_1	5	3
a_1	b_2	6	7
a_2	b_3	8	10
a_2	b_3	8	2

连接(续)

- 外连接
 - 如果把舍弃的元组也保存在结果关系中，而在其他属性上填空值(Null)，这种连接就叫做外连接（**OUTER JOIN**）。
- 左外连接
 - 如果只把左边关系 R 中要舍弃的元组保留就叫做左外连接(**LEFT OUTER JOIN**或**LEFT JOIN**)
- 右外连接
 - 如果只把右边关系 S 中要舍弃的元组保留就叫做右外连接(**RIGHT OUTER JOIN**或**RIGHT JOIN**)。

连接(续)

下图是例5中关系 R 和关系 S 的外连接

A	B	C	E
a_1	b_1	5	3
a_1	b_2	6	7
a_2	b_3	8	10
a_2	b_3	8	2
a_2	b_4	12	NULL
NULL	b_5	NULL	2

(a) 外连接

连接(续)

图(b)是例5中关系 R 和关系 S 的左外连接,图(c)是右外连接

A	B	C	E
a_1	b_1	5	3
a_1	b_2	6	7
a_2	b_3	8	10
a_2	b_3	8	2
a_2	b_4	12	NULL

(b) 左外连接

A	B	C	E
a_1	b_1	5	3
a_1	b_2	6	7
a_2	b_3	8	10
a_2	b_3	8	2
NULL	b_5	NULL	2

(c) 右外连接

除(续)

[例6]设关系 R 、 S 分别为下图的(a)和(b), $R \div S$ 的结果为图(c)

R		
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

(a)

S		
B	C	D
b_1	c_2	d_1
b_2	c_1	d_1
b_2	c_3	d_2

(b)

$R \div S$
A
a_1

(c)

分析

- 在关系R中，A可以取四个值{a1, a2, a3, a4}
 a_1 的象集为 $\{(b_1, c_2), (b_2, c_3), (b_2, c_1)\}$
 a_2 的象集为 $\{(b_3, c_7), (b_2, c_3)\}$
 a_3 的象集为 $\{(b_4, c_6)\}$
 a_4 的象集为 $\{(b_6, c_6)\}$
 - S在(B, C)上的投影为
 $\{(b1, c2), (b2, c1), (b2, c3)\}$
 - 只有 a_1 的象集包含了S在(B, C)属性组上的投影
- 所以 $R \div S = \{a_1\}$

3.1 SQL概述

- SQL (Structured Query Language)

结构化查询语言，是关系数据库的标准语言

- SQL是一个通用的、功能极强的关系数据库语言

3.1.2 SQL的特点

1.综合统一

- 集数据定义语言（DDL），数据操纵语言（DML），数据控制语言（DCL）功能于一体。
- 可以独立完成数据库生命周期中的全部活动：
 - 定义关系模式，插入数据，建立数据库；
 - 对数据库中的数据进行查询和更新；
 - 数据库重构和维护
 - 数据库安全性、完整性控制等
- 用户数据库投入运行后，可根据需要随时逐步修改模式，不影响数据的运行。
- 数据操作符统一

2.高度非过程化

- 非关系数据模型的数据操纵语言“面向过程”，必须制定存取路径
- SQL只要提出“做什么”，无须了解存取路径。
- 存取路径的选择以及SQL的操作过程由系统自动完成。

3.面向集合的操作方式

- 非关系数据模型采用面向记录的操作方式，操作对象是一条记录
- **SQL**采用集合操作方式
 - 操作对象、查找结果可以是元组的集合
 - 一次插入、删除、更新操作的对象可以是元组的集合

4.以同一种语法结构提供多种使用方式

- SQL是独立的语言

能够独立地用于联机交互的使用方式

- SQL又是嵌入式语言

SQL能够嵌入到高级语言（例如C，C++，Java）程序中，供程序员设计程序时使用

5.语言简洁，易学易用

- SQL功能极强，完成核心功能只用了9个动词。

表 3.1 SQL 语言的动词

SQL 功 能	动 词
数 据 查 询	SELECT
数 据 定 义	CREATE, DROP, ALTER
数 据 操 纵	INSERT, UPDATE DELETE
数 据 控 制	GRANT, REVOKE

数据查询

- 语句格式

```
SELECT [ALL|DISTINCT] <目标列表达式>  
                                     [, <目标列表达式>] ...  
  
FROM <表名或视图名>[, <表名或视图名> ] ...  
  
[ WHERE <条件表达式> ]  
  
[ GROUP BY <列名1> [ HAVING <条件表达式> ] ]  
  
[ ORDER BY <列名2> [ ASC|DESC ] ];
```

一、插入元组

- 语句格式

INSERT

INTO <表名> [(<属性列1>[, <属性列2 >...])]

VALUES (<常量1> [, <常量2>] ...)

- 功能

- 将新元组插入指定表中

插入元组（续）

- **INTO子句**
 - 属性列的顺序可与表定义中的顺序不一致
 - 没有指定属性列
 - 指定部分属性列
- **VALUES子句**
 - 提供的值必须与**INTO**子句匹配
 - 值的个数
 - 值的类型

插入元组（续）

[例1] 将一个新学生元组（学号：200215128；姓名：陈冬；性别：男；所在系：IS；年龄：18岁）插入到 Student表中。

INSERT

INTO Student (Sno, Sname, Ssex, Sdept, Sage)

VALUES ('200215128', '陈冬', '男', 'IS', 18);

插入元组（续）

[例2] 将学生张成民的信息插入到Student表中。

```
INSERT  
INTO Student  
VALUES ('200215126', '张成民', '男', 18,  
'CS');
```


插入元组（续）

[例3] 插入一条选课记录('200215128', '1 ')。

```
INSERT
```

```
INTO SC(Sno, Cno)
```

```
VALUES ( ' 200215128 ', ' 1 ' );
```

RDBMS将在新插入记录的**Grade**列上自动地赋空值。

或者：

```
INSERT
```

```
INTO SC
```

```
VALUES ( ' 200215128 ', ' 1 ', NULL);
```

3.4.2 修改数据

- 语句格式

UPDATE <表名>

SET <列名>=<表达式>[, <列名>=<表达式>]...

[WHERE <条件>];

- 功能

- 修改指定表中满足WHERE子句条件的元组

1. 修改某一个元组的值

[例5] 将学生200215121的年龄改为22岁

```
UPDATE Student
```

```
SET Sage=22
```

```
WHERE Sno=' 200215121 ';
```

修改数据（续）

- 三种修改方式
 1. 修改某一个元组的值
 2. 修改多个元组的值
 3. 带子查询的修改语句

3.5.3 删除数据

- 语句格式
DELETE
FROM <表名>
[WHERE <条件>];
- 功能
 - 删除指定表中满足**WHERE**子句条件的元组
- **WHERE**子句
 - 指定要删除的元组
 - 缺省表示要删除表中的全部元组，表的定义仍在字典中

3.3.3 索引的建立与删除

- 建立索引的目的：加快查询速度
- 谁可以建立索引
 - DBA 或 表的属主（即建立表的人）
 - DBMS一般会建立以下列上的索引

PRIMARY KEY

UNIQUE

- 谁 维护索引
 - DBMS自动完成
- 使用索引
 - DBMS自动选择是否使用索引以及使用哪些索引

3.6 视 图

视图的特点

- 虚表，是从一个或几个基本表（或视图）导出的表
- 只存放视图的定义，不存放视图对应的数据
- 基表中的数据发生变化，从视图中查询出的数据也随之改变

一、建立视图

- 语句格式

CREATE VIEW

<视图名> [(<列名> [, <列名>]...)]

AS <子查询>

[WITH CHECK OPTION];

- 组成视图的属性列名：全部省略或全部指定
- 子查询不允许含有**ORDER BY**子句和**DISTINCT**(SQL Server允许Distinct)短语

3.6.4 视图的作用

- 1. 视图能够简化用户的操作
- 2. 视图使用户能以多种角度看待同一数据
- 3. 视图对重构数据库提供了一定程度的逻辑独立性
- 4. 视图能够对机密数据提供安全保护
- 5. 适当的利用视图可以更清晰的表达查询

关系数据库规范化理论

- 如何设计一个适合的关系数据库系统，关键是关系数据库模式的设计，一个好的关系数据库模式应该包括多少关系模式，而每一个关系模式又应该包括哪些属性，又如何将这些相互关联的关系模式组建一个适合的关系模型，这些工作属于数据库设计的问题，确切地讲是数据库逻辑设计的问题，有关数据库设计的全过程将在第4章详细讨论
- 本章讲述关系数据库规范化理论，这是数据库逻辑设计的理论依据。
 - 要求了解规范化理论的研究动机及其在数据库设计中的作用，
 - 掌握函数依赖的有关概念，
 - 第一范式、第二范式、第三范式的定义，
 - 重点掌握并能够灵活运用关系模式规范化的方法和关系模式分解的方法，这也是本章的难点。

规范化理论内容

- 关系数据库的规范化理论主要包括三个方面的内容：
 - 函数依赖
 - 范式 (Normal Form)
 - 模式设计
- 其中，函数依赖起着核心的作用，是模式分解和模式设计的基础，范式是模式分解的标准。

关系模式的存储异常问题

- 一个好的关系模式应该具备以下四个条件：
 - 1. 尽可能少的数据冗余。
 - 2. 没有插入异常。
 - 3. 没有删除异常。
 - 4. 没有更新异常。

6.2 规范化

规范化理论正是用来改造关系模式，通过分解关系模式来消除其中不合适的数据依赖，以解决插入异常、删除异常、更新异常和数据冗余问题。

1. 删除某一个元组的值

[例8] 删除学号为200215128的学生记录。

```
DELETE  
FROM Student  
WHERE Sno= 200215128 ';
```

关系模式的存储异常问题

- 如何按照一定的规范设计关系模式，将结构复杂的关系分解成结构简单的关系，降低或消除数据库中冗余数据的过程，这就是关系的规范化。

6.2 规范化

6.2.1 函数依赖

6.2.2 码

6.2.3 范式

6.2.4 2NF

6.2.5 3NF

6.2.6 BCNF

6.2.1 函数依赖

- 函数依赖
- 平凡函数依赖与非平凡函数依赖
- 完全函数依赖与部分函数依赖
- 传递函数依赖

一、函数依赖

定义**6.1** 设 $R(U)$ 是一个属性集 U 上的关系模式， X 和 Y 是 U 的子集。

若对于 $R(U)$ 的任意一个可能的关系 r ， r 中不可能存在两个元组在 X 上的属性值相等，而在 Y 上的属性值不等，则称“ X 函数确定 Y ”或“ Y 函数依赖于 X ”，记作 $X \rightarrow Y$ 。

二、平凡函数依赖与非平凡函数依赖

在关系模式 $R(U)$ 中，对于 U 的子集 X 和 Y ，

如果 $X \rightarrow Y$ ，但 $Y \subsetneq X$ ，则称 $X \rightarrow Y$ 是非平凡的函数依赖

若 $X \rightarrow Y$ ，但 $Y \subseteq X$ ，则称 $X \rightarrow Y$ 是平凡的函数依赖

- 例：在关系 $SC(Sno, Cno, Grade)$ 中，

非平凡函数依赖： $(Sno, Cno) \rightarrow Grade$

平凡函数依赖： $(Sno, Cno) \rightarrow Sno$

$(Sno, Cno) \rightarrow Cno$

平凡函数依赖与非平凡函数依赖（续）

- 若 $X \rightarrow Y$ ，则 X 称为这个函数依赖的决定属性组，也称为决定因素（**Determinant**）。
- 若 $X \rightarrow Y$ ， $Y \rightarrow X$ ，则记作 $X \leftrightarrow Y$ 。
- 若 Y 不函数依赖于 X ，则记作 $X \nrightarrow Y$ 。

完全函数依赖与部分函数依赖（续）

[例1] 中 $(Sno, Cno) \xrightarrow{F} Grade$ 是完全函数依赖,

$(Sno, Cno) \xrightarrow{P} Sdept$ 是部分函数依赖

因为 $Sno \rightarrow Sdept$ 成立, 且 Sno 是 (Sno, Cno) 的真子集

四、传递函数依赖

定义**6.3** 在 $R(U)$ 中, 如果 $X \rightarrow Y$, $(Y \subsetneq X)$, $Y \not\rightarrow X$ $Y \rightarrow Z$,
则称 Z 对 X 传递函数依赖。
记为: $X \xrightarrow{\text{传递}} Z$

注: 如果 $Y \rightarrow X$, 即 $X \longleftrightarrow Y$, 则 Z 直接依赖于 X 。

例: 在关系 $Std(Sno, Sdept, Mname)$ 中, 有:

$Sno \rightarrow Sdept$, $Sdept \rightarrow Mname$

$Mname$ 传递函数依赖于 Sno

6.2.2 码

定义**6.4** 设K为R<U,F>中的属性或属性组合。若 $K \xrightarrow{F} U$,
则K称为R的**候选码**（Candidate Key）。


若候选码多于一个，则选定其中的一个做为**主码**
（Primary Key）。

码（续）

- 主属性与非主属性
 - 包含在任何一个候选码中的属性，称为主属性（**Prime attribute**）
 - 不包含在任何码中的属性称为非主属性（**Nonprime attribute**）或非码属性（**Non-key attribute**）
- 全码
 - 整个属性组是码，称为全码（**All-key**）

6.2.3 范式

- 范式是符合某一种级别的关系模式的集合
- 关系数据库中的关系必须满足一定的要求。满足不同程度要求的为不同范式
- 范式的种类：



- 第一范式(1NF)
- 第二范式(2NF)
- 第三范式(3NF)
- BC范式(BCNF)
- 第四范式(4NF)
- 第五范式(5NF)

6.2.3 范式

- 各种范式之间存在联系：

$$1NF \supset 2NF \supset 3NF \supset BCNF \supset 4NF \supset 5NF$$

- 某一关系模式R为第n范式，可简记为 $R \in nNF$ 。
- 一个低一级范式的关系模式，通过模式分解可以转换为若干个高一级范式的关系模式的集合，这种过程就叫规范化

2NF (续)

[例4] 关系模式 S-L-C(Sno, Sdept, Sloc, Cno, Grade)

Sloc为学生住处，假设每个系的学生住在同一个地方

- 函数依赖包括：

$(Sno, Cno) \xrightarrow{F} Grade$

$Sno \rightarrow Sdept$

$(Sno, Cno) \xrightarrow{P} Sdept$

$Sno \xrightarrow{\overline{}} Sloc$

$(Sno, Cno) \xrightarrow{P} Sloc$

$Sdept \rightarrow Sloc$

2NF (续)

- 2NF的定义

定义**6.6** 若 $R \in 1NF$ ，且每一个非主属性完全函数依赖于码，则 $R \in 2NF$ 。

例：S-L-C(Sno, Sdept, Sloc, Cno, Grade) $\in 1NF$

S-L-C(Sno, Sdept, Sloc, Cno, Grade) $\in 2NF$

SC (Sno, Cno, Grade) $\in 2NF$

S-L (Sno, Sdept, Sloc) $\in 2NF$

6.2.5 3NF

- 3NF的定义

定义**6.7** 关系模式 $R\langle U, F \rangle$ 中若不存在这样的码 X 、属性组 Y 及非主属性 Z ($Z \not\subseteq Y$), 使得 $X \rightarrow Y$, $Y \rightarrow Z$ 成立, $Y \not\rightarrow X$, 则称 $R\langle U, F \rangle \in 3NF$ 。

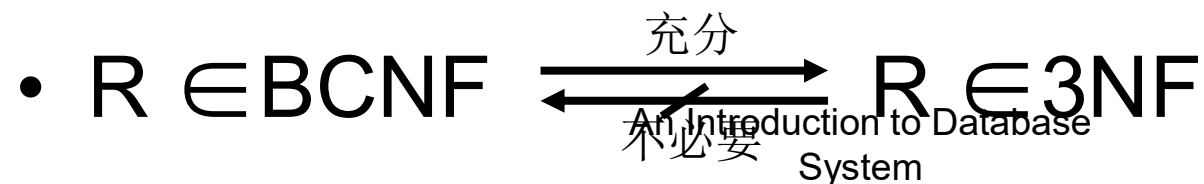
■若 $R \in 3NF$, 则每一个非主属性既不部分依赖于码也不传递依赖于码。

6.2.6 BC范式 (BCNF)

- 定义**6.8** 关系模式 $R\langle U, F \rangle \in 1NF$, 若 $X \rightarrow Y$ 且 $Y \subseteq X$ 时 X 必含有码, 则 $R\langle U, F \rangle \in BCNF$ 。
- 等价于: 每一个决定属性因素都包含码

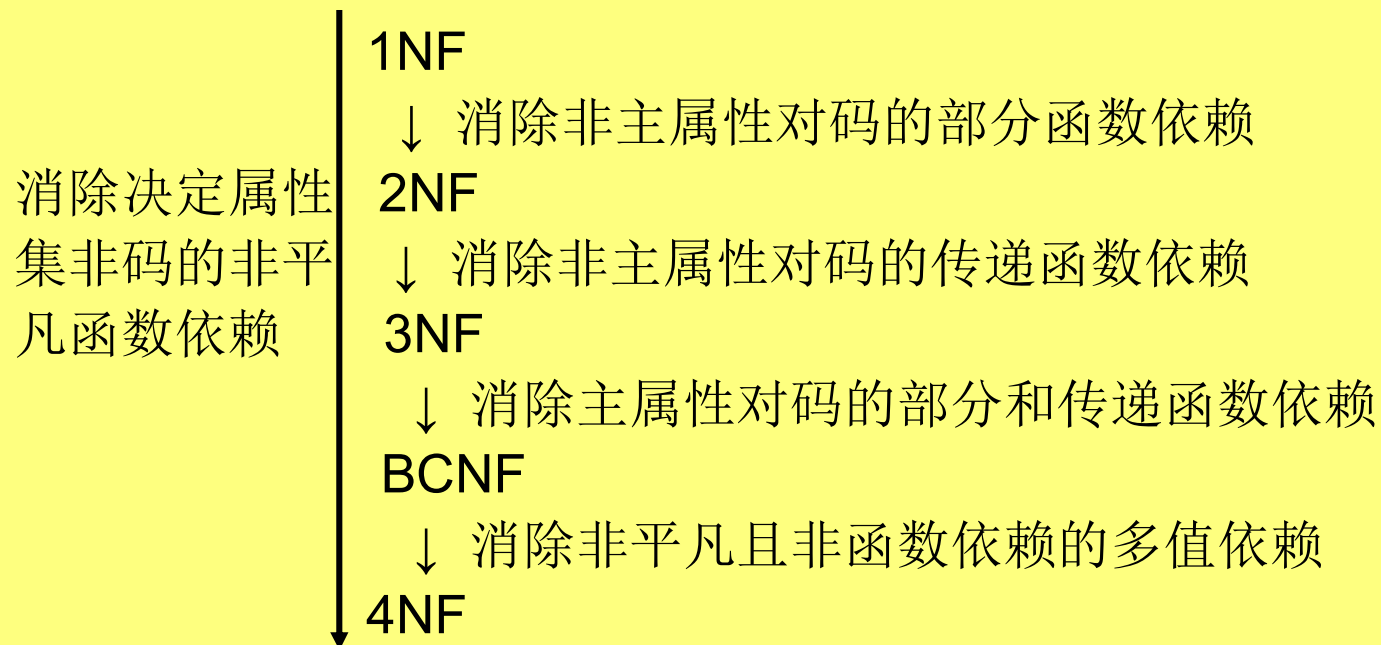
BCNF (续)

- 若 $R \in \text{BCNF}$
 - 所有非主属性对每一个码都是完全函数依赖
 - 所有的主属性对每一个不包含它的码，也是完全函数依赖
 - 没有任何属性完全函数依赖于非码的任何一组属性



规范化小结（续）

- 关系模式规范化的基本步骤



6.3 数据依赖的公理系统

- 逻辑蕴含

定义6.11 对于满足一组函数依赖 F 的关系模式 $R \langle U, F \rangle$, 其任何一个关系 r , 若函数依赖 $X \rightarrow Y$ 都成立, (即 r 中任意两元组 t, s , 若 $t[X] = s[X]$, 则 $t[Y] = s[Y]$), 则称 F 逻辑蕴含 $X \rightarrow Y$

1. Armstrong公理系统

关系模式 $R \langle U, F \rangle$ 来说有以下的推理规则：

- A1. 自反律 (Reflexivity) : 若 $Y \subseteq X \subseteq U$, 则 $X \rightarrow Y$ 为 F 所蕴含。
- A2. 增广律 (Augmentation) : 若 $X \rightarrow Y$ 为 F 所蕴含, 且 $Z \subseteq U$, 则 $XZ \rightarrow YZ$ 为 F 所蕴含。
- A3. 传递律 (Transitivity) : 若 $X \rightarrow Y$ 及 $Y \rightarrow Z$ 为 F 所蕴含, 则 $X \rightarrow Z$ 为 F 所蕴含。

2. 导出规则

1. 根据A1, A2, A3这三条推理规则可以得到下面三条推理规则:

— 合并规则: 由 $X \rightarrow Y$, $X \rightarrow Z$, 有 $X \rightarrow YZ$ 。

(A2, A3)

— 伪传递规则: 由 $X \rightarrow Y$, $WY \rightarrow Z$, 有 $XW \rightarrow Z$ 。

(A2, A3)

— 分解规则: 由 $X \rightarrow Y$ 及 $Z \subseteq Y$, 有 $X \rightarrow Z$ 。

(A1, A3)

导出规则

2.根据合并规则和分解规则，可得引理6.1

引理6.1 $X \rightarrow A_1 A_2 \dots A_k$ 成立的充分必要条件是 $X \rightarrow A_i$ 成立 ($i=1, 2, \dots, k$)

3. 函数依赖闭包

定义6.12 在关系模式 $R\langle U, F \rangle$ 中为 F 所逻辑蕴含的函数依赖的全体叫作 F 的闭包，记为 F^+ 。

定义6.13 设 F 为属性集 U 上的一组函数依赖， $X \subseteq U$ ， $X_F^+ = \{ A | X \rightarrow A \text{ 能由 } F \text{ 根据Armstrong公理导出} \}$ ， X_F^+ 称为属性集 X 关于函数依赖集 F 的闭包

关于闭包的引理

- 引理**6.2**

设 F 为属性集 U 上的一组函数依赖, $X, Y \subseteq U$, $X \rightarrow Y$ 能由 F 根据Armstrong公理导出的充分必要条件是 $Y \subseteq X_F^+$

- 用途

将判定 $X \rightarrow Y$ 是否能由 F 根据Armstrong公理导出的问题, 转化为求出 X_F^+ 、判定 Y 是否为 X_F^+ 的子集的问题

求闭包的算法

算法6.1 求属性集 X ($X \subseteq U$) 关于 U 上的函数依赖集 F 的闭包 X_F^+

输入: X, F

输出: X_F^+

步骤:

(1) 令 $X^{(0)} = X, i=0$

(2) 求 B , 这里 $B = \{ A | (\exists V)(\exists W)(V \rightarrow W \in F \wedge V \subseteq X^{(i)} \wedge A \in W) \}$;

(3) $X^{(i+1)} = B \cup X^{(i)}$

(4) 判断 $X^{(i+1)} = X^{(i)}$ 吗?

(5) 若相等或 $X^{(i)} = U$, 则 $X^{(i)}$ 就是 X_F^+ , 算法终止。

(6) 若否, 则 $i=i+1$, 返回第 (2) 步。

函数依赖闭包

[例1] 已知关系模式 $R\langle U, F\rangle$, 其中

$$U=\{A, B, C, D, E\};$$

$$F=\{AB\rightarrow C, B\rightarrow D, C\rightarrow E, EC\rightarrow B, AC\rightarrow B\}。$$

求 $(AB)_F^+$ 。

解 设 $X^{(0)}=AB$;

$$(1) X^{(1)}=AB\cup CD=ABCD。$$

$$(2) X^{(0)} \neq X^{(1)}$$

$$X^{(2)}=X^{(1)}\cup BE=ABCDE。$$

$$(3) X^{(2)}=U, \text{ 算法终止}$$

$$\rightarrow (AB)_F^+=ABCDE。$$

$R < U, F >, U = (A, B, C, G, H, I), F = \{A \rightarrow B, A \rightarrow C, CG \rightarrow H, CG \rightarrow I, B \rightarrow H\}$, 计算 $(AG)_F^+$ 。

所用依赖

(1) $X^{(0)} = AG$

(2) $A \rightarrow B$

$A \rightarrow C$

$X^{(1)} = AGBC$

(3) $CG \rightarrow H$

$CG \rightarrow I$

$X^{(2)} = AGBCH I$

如果 $(X)_F = U$,
那么可以考虑
它是不是候选
码, 如何考虑?

求关键字问题可转化为求属性集闭包问题,

因为 $X \rightarrow X^+$, 若 $U = X^+$, 则 $X \rightarrow U$, 即 X 为关键字。

例：求关系模式 $R=(ABCD, \{A \rightarrow B, B \rightarrow C\})$ 的关键字

验证： $(AD)^+ = ABCD$

- 设有关系模式 $R(U, F)$ ，其中： $U=\{A, B, C, D, E, P\}$ ， $F=\{A \rightarrow B, C \rightarrow P, E \rightarrow A, CE \rightarrow D\}$ ，求出 R 的所有候选关键字。
- 解：根据候选关键字的定义：如果函数依赖 $X \rightarrow U$ 在 R 上成立，且不存在任何 $X' \subseteq X$ ，使得 $X' \rightarrow U$ 也成立，则称 X 是 R 的一个候选关键字。由此可知，候选关键字只可能由 A, C, E 组成，但有 $E \rightarrow A$ ，所以组成候选关键字的属性可能是 CE 。
计算可知： $(CE)^+ = ABCDEP$ ，即 $CE \rightarrow U$ 。而：
 $C^+ = CP$ ， $E^+ = ABE$
故 R 只有一个候选关键字 CE 。

- 设有关系模式 $R(C, T, S, N, G)$ ，其上的函数依赖集： $F=\{C \rightarrow T, CS \rightarrow G, S \rightarrow N\}$ ，求出 R 的所有候选关键字。
- 解：根据候选关键字的定义， R 的候选关键字只可能由 F 中各个函数依赖的左边属性组成，即 C, S ，所以组成候选关键字的属性可能是 CS 。
计算可知： $(CS)^+ = CGNST$ ，即 $CS \rightarrow U$ 。而：
 $C^+ = CT$ ， $S^+ = NS$ 。故 R 只有一个候选关键字 CS 。

- 设有关系模式R (C, T, S, N, G, **Z**)，其上的函数依赖集： $F=\{C\rightarrow T, CS\rightarrow G, S\rightarrow N\}$ ，求出R的所有候选关键字。

6. 最小依赖集

定义6.15 如果函数依赖集 F 满足下列条件，则称 F 为一个极小函数依赖集。亦称为最小依赖集或最小覆盖。

(1)单属性： F 中任一函数依赖的右部仅含有一个属性。

(2)无冗余化： F 中不存在这样的函数依赖 $X \rightarrow A$ ，使得 F 与 $F - \{X \rightarrow A\}$ 等价。

(3) 既约化： F 中不存在这样的函数依赖 $X \rightarrow A$ ， X 有真子集 Z 使得 $F - \{X \rightarrow A\} \cup \{Z \rightarrow A\}$ 与 F 等价。

极小化过程（续）

- (1) 逐一检查 F 中各函数依赖 $FD_i: X \rightarrow Y$, 若 $Y = A_1 A_2 \dots A_k$, $k > 2$, 则用 $\{X \rightarrow A_j | j=1, 2, \dots, k\}$ 来取代 $X \rightarrow Y$ 。
- (2) 逐一检查 F 中各函数依赖 $FD_i: X \rightarrow A$, 令 $G = F - \{X \rightarrow A\}$, 若 $A \in X_G^+$, 则从 F 中去掉此函数依赖。
- (3) 逐一取出 F 中各函数依赖 $FD_i: X \rightarrow A$, 设 $X = B_1 B_2 \dots B_m$, 逐一考查 B_i ($i=1, 2, \dots, m$), 若 $A \in (X - B_i)_F^+$, 则以 $X - B_i$ 取代 X 。

极小化过程（续）

[例3] $F = \{A \rightarrow B, B \rightarrow A, B \rightarrow C, A \rightarrow C, C \rightarrow A\}$

F_{m1} 、 F_{m2} 都是 F 的最小依赖集:

$$F_{m1} = \{A \rightarrow B, B \rightarrow C, C \rightarrow A\}$$

$$F_{m2} = \{A \rightarrow B, B \rightarrow A, A \rightarrow C, C \rightarrow A\}$$

- F 的最小依赖集 F_m 不唯一
- 极小化过程(定理6.3的证明)也是检验 F 是否为极小依赖集的一个算法

$F = \{C \rightarrow A, A \rightarrow G, CG \rightarrow B, B \rightarrow A\}$, 求 F_c 。

F是无冗余的。

判断 $CG \rightarrow B$,

$$(CG - C)_F^+ == (G)_F^+ = \{G\}$$

$$B \notin (CG - C)_F^+$$

$$(CG - G)_F^+ = (C)_F^+ = \{CAGB\}$$

$$B \in (CG - G)_F^+, \text{ 以 } C \text{ 代替 } CG$$

最后, $F_c = \{C \rightarrow A, A \rightarrow G, C \rightarrow B, B \rightarrow A\}$

6.4 模式的分解

- 把低一级的关系模式分解为若干个高一级的关系模式的方法不是唯一的
- 只有能够保证分解后的关系模式与原关系模式等价，分解方法才有意义

关系模式分解的标准

三种模式分解等价的定义：

1. 分解具有无损连接性
2. 分解要保持函数依赖
3. 分解既要保持函数依赖，又要具有无损连接性

具有无损连接性的模式分解

- 关系模式 $R\langle U, F \rangle$ 的一个分解 $\rho = \{ R_1\langle U_1, F_1 \rangle, R_2\langle U_2, F_2 \rangle, \dots, R_n\langle U_n, F_n \rangle \}$

若 R 与 R_1 、 R_2 、...、 R_n 自然连接的结果相等，则称关系模式 R 的这个分解 ρ 具有无损连接性（Lossless join）

- 具有无损连接性的分解保证不丢失信息
- 无损连接性不一定能解决插入异常、删除异常、修改复杂、数据冗余等问题

保持函数依赖的模式分解

设关系模式 $R\langle U, F \rangle$ 被分解为若干个关系模式

$$R_1\langle U_1, F_1 \rangle, R_2\langle U_2, F_2 \rangle, \dots, R_n\langle U_n, F_n \rangle$$

（其中 $U = U_1 \cup U_2 \cup \dots \cup U_n$ ，且不存在 $U_i \subseteq U_j$ ， F_i 为 F 在 U_i 上的投影），若 F 所逻辑蕴含的函数依赖一定也由分解得到的某个关系模式中的函数依赖 F_i 所逻辑蕴含，则称关系模式 R 的这个分解是保持函数依赖的（**Preserve dependency**）

分解算法

- 算法6.2 判别一个分解的无损连接性
- 算法6.3（合成法）转换为**3NF**的保持函数依赖的分解。

– **算法：**（判别一个分解的无损连接性）

$U = \{A_1, A_2, \dots, A_n\}$

$\rho = \{R_1 \langle U_1, F_1 \rangle, R_2 \langle U_2, F_2 \rangle, \dots, R_k \langle U_k, F_k \rangle\}$

ρ **注意：F要是最小覆盖**

1. 建立一个n列k行的矩阵

– $TB = \{C_{ij} \mid \text{若 } A_j \in U_i, C_{ij} = a_j, \text{ 否则 } C_{ij} = b_{ij}\}$

	A_1	\dots	A_n
U_1			
\dots		C_{ij}	
U_k			

An Introduction to Database System

判别一个分解的无损连接性(2)

- 2. 对 F 中每一个函数依赖 $X \rightarrow Y$, 若 TB 中存在元组 t_1, t_2 , 使得 $t_1[X] = t_2[X]$, $t_1[Y] \neq t_2[Y]$, 则对每一个 $A_i \in Y$:
 - ① 若 $t_1[A_i], t_2[A_i]$ 中有一个等于 a_j , 则另一个也改为 a_j ;
 - ② 若 ① 不成立, 则取 $t_2[A_i] = t_1[A_i]$ (t_2 的行号小于 t_1)。

判别一个分解的无损连接性(3)

3.反复执行2 , 3 , 直至:

- ①TB中出现一行为 a_1, a_2, \dots, a_n 的一行。
- ②TB不再发生变化, 且没有一行为 a_1, \dots, a_n 。

在①情况下, ρ 为无损分解, 否则为有损分解。

示例一： $U=\{A,B,C,D,E\}$, $F=\{AB\rightarrow C, C\rightarrow D,D\rightarrow E\}$
 $\rho =\{(A, B, C), (C, D), (D, E)\}$

第一步:

	A	B	C	D	E
ABC	a_1	a_2	a_3	b_{14}	b_{15}
CD	b_{21}	b_{22}	a_3	a_4	b_{25}
DE	b_{31}	b_{32}	b_{33}	a_4	a_5

示例一： $U=\{A,B,C,D,E\}$, $F=\{AB\rightarrow C, C\rightarrow D, D\rightarrow E\}$
 $\rho =\{(A, B, C), (C, D), (D, E)\}$

第二步: $AB\rightarrow C$

	A	B	C	D	E
ABC	a_1	a_2	a_3	b_{14}	b_{15}
CD	b_{21}	b_{22}	a_3	a_4	b_{25}
DE	b_{31}	b_{32}	b_{33}	a_4	a_5

示例一： $U=\{A,B,C,D,E\}$, $F=\{AB\rightarrow C, C\rightarrow D, D\rightarrow E\}$

$\rho =\{(A, B, C), (C, D), (D, E)\}$

第三步: $C\rightarrow D$

	A	B	C	D	E
ABC	a_1	a_2	a_3	a_4	b_{15}
CD	b_{21}	b_{22}	a_3	a_4	b_{25}
DE	b_{31}	b_{32}	b_{33}	a_4	a_5

示例一： $U=\{A,B,C,D,E\}$, $F=\{AB\rightarrow C, C\rightarrow D, D\rightarrow E\}$

$\rho =\{(A, B, C), (C, D), (D, E)\}$

第四步: $D\rightarrow E$

	A	B	C	D	E
ABC	a_1	a_2	a_3	a_4	a_5
CD	b_{21}	b_{22}	a_3	a_4	a_5
DE	b_{31}	b_{32}	b_{33}	a_4	a_5

关系模式的分解算法

示例一： $U=\{A,B,C,D,E\}$, $F=\{AB\rightarrow C, C\rightarrow D, D\rightarrow E\}$

$AB\rightarrow C$

	A	B	C	D	E
ABC	a_1	a_2	a_3	b_{14}	b_{15}
CD	b_{21}	b_{22}	a_3	a_4	b_{25}
DE	b_{31}	b_{32}	b_{33}	a_4	a_5

	A	B	C	D	E
ABC	a_1	a_2	a_3	b_{14}	b_{15}
CD	b_{21}	b_{22}	a_3	a_4	b_{25}
DE	b_{31}	b_{32}	b_{33}	a_4	a_5

$C\rightarrow D$

	A	B	C	D	E
ABC	a_1	a_2	a_3	a_4	b_{15}
CD	b_{21}	b_{22}	a_3	a_4	b_{25}
DE	b_{31}	b_{32}	b_{33}	a_4	a_5

$D\rightarrow E$

	A	B	C	D	E
ABC	a_1	a_2	a_3	a_4	a_5
CD	b_{21}	b_{22}	a_3	a_4	a_5
DE	b_{31}	b_{32}	b_{33}	a_4	a_5

$\rho = \{(A, B, C), (C, D), (D, E)\}$

例：设关系模式R（ABCD），R分解成 $\rho=\{AB,BC,CD\}$ 。如果R上成立的函数依赖集 $F1=\{B \rightarrow A, C \rightarrow D\}$,那么 ρ 相对于F1是否无损分解？如果R上成立的函数依赖集 $F2=\{A \rightarrow B, C \rightarrow D\}$ 呢？

初始

	A	B	C	D
AB	a1	a2	b13	b14
BC	b21	a2	a3	b24
CD	b31	b32	a3	a4

B→A

	A	B	C	D
AB	a1	a2	b13	b14
BC	a1	a2	a3	b24
CD	b31	b32	a3	a4

C→D

	A	B	C	D
AB	a1	a2	b13	b14
BC	a1	a2	a3	a4
CD	b31	b32	a3	a4

初始

	A	B	C	D
AB	a1	a2	b13	b14
BC	b21	a2	a3	b24
CD	b31	b32	a3	a4

A→B

	A	B	C	D
AB	a1	a2	b13	b14
BC	b21	a2	a3	b24
CD	b31	b32	a3	a4

C→D

	A	B	C	D
AB	a1	a2	b13	b14
BC	b21	a2	a3	a4
CD	b31	b32	a3	a4

例题：

设有关系模式 $R(U, P)$ ，其中： $U=\{A, B, C, D, E\}$ ， $F=\{A \rightarrow D, E \rightarrow D, D \rightarrow B, BC \rightarrow D, DC \rightarrow A\}$

(1) 求出 R 的候选关键字。

(2) 判断 $\rho=\{AB, AE, CE, BCD, AC\}$ 是否为无损连接分解？

初判： A, E, D, B, C ，但 D, B, A 出现在右边，所以只剩下： E 和 C ，求 $(EC)_{F^+} = \{ECDBA\} = U$

$(E)_{F^+} = \{EDB\}$ $(C)_{F^+} = \{C\}$

$F = F_c$? $DC \rightarrow A$ 是冗余的

$F_c = \{A \rightarrow D, E \rightarrow D, D \rightarrow B, BC \rightarrow D\}$

	A	B	C	D	E
AB	a1	a2	b13	b14	b15
AE	a1	b22	b23	b24	a5
CE	b31	b32	a3	b34	a5
BCD	b41	a2	a3	a4	b45
AC	a1	b52	a3	b54	b55

A->D

	A	B	C	D	E
AB	a1	a2	b13	b14	b15
AE	a1	b22	b23	b14	a5
CE	b31	b32	a3	b34	a5
BCD	b41	a2	a3	a4	b45
AC	a1	b52	a3	b14	b55

E->D

	A	B	C	D	E
AB	a1	a2	b13	b14	b15
AE	a1	b22	b23	b14	a5
CE	b31	b32	a3	b14	a5
BCD	b41	a2	a3	a4	b45
AC	a1	b52	a3	b14	b55

D->B

	A	B	C	D	E
AB	a1	a2	b13	b14	b15
AE	a1	a2	b23	b14	a5
CE	b31	a2	a3	b14	a5
BCD	b41	a2	a3	a4	b45
AC	a1	a2	a3	b14	b55

BC->D

	A	B	C	D	E
AB	a1	a2	b13	b14	b15
AE	a1	a2	b23	b14	a5
CE	b31	a2	a3	a4	a5
BCD	b41	a2	a3	a4	b45
AC	a1	a2	a3	a4	b55

An Introduction to Database
System

算法：（达到3NF且保持函数依赖的分解）

- 1 求F的正则覆盖 F_C 。
- 2 找出不在 F_C 中出现的属性，将它们构成一个关系模式，并从U中去掉它们(剩余属性仍记为U)。
- 3 若有 $X \rightarrow A \in F_C$ ，且 $XA=U$ ，则 $\rho=\{R\}$ ，算法终止。
- 4 对 F_C 按具有相同左部的原则进行分组（设为k组），每一组函数依赖所涉及的属性全体为 U_i ，令 F_i 为 F_C 在 U_i 上的投影，则 $\rho = \{R_1 \langle U_1, F_1 \rangle, \dots, R_k \langle U_k, F_k \rangle\}$ 是 $R \langle U, F \rangle$ 的一个保持函数依赖的分解，并且每个 $R_i \langle U_i, F_i \rangle \in 3NF$ 。

– 示例

– $U = \{S\#, SD, MN, C\#, G\}$

– $F = \{S\# \rightarrow SD, S\# \rightarrow MN, SD \rightarrow MN, (S\#, C\#) \rightarrow G\}$

– 1 $F_c = \{S\# \rightarrow SD, SD \rightarrow MN, (S\#, C\#) \rightarrow G\}$

– 2 分组

– $\{(S\#, SD), S\# \rightarrow SD\}$

– $\{(SD, MN), SD \rightarrow MN\}$

– $\{(S\#, C\#, G), (S\#, C\#) \rightarrow G\}$

- 设有关系模式 $R(U, F)$ ，其中： $U=\{C, T, H, R, S, G\}$ ， $F=\{C \rightarrow R, S \rightarrow R, R \rightarrow T, TH \rightarrow R, RH \rightarrow C\}$ ，将其保持依赖性分解为3NF。

首先判断 F 是否最小覆盖

- ❖ $R_0=G$ $R_1=CR$, $R_2=SR$, $R_3=RT$, $R_4=THR$, $R_5=RHC$
- ❖ $R_1 \in R_5$ $R_3 \in R_4$
- ❖ $\rho=\{R_2(S, R), R_4(T, H, R), R_5(R, H, C), G\}$

第七章 数据库设计

7.1 数据库设计概述

7.2 需求分析

7.3 概念结构设计

7.4 逻辑结构设计

7.5 数据库的物理设计

7.6 数据库实施和维护

7.7 小结

An Introduction to Database
System

7.2.3 数据字典

- 数据字典的用途
 - 进行详细的数据收集和数据分析所获得的主要结果
- 数据字典的内容
 - 数据项
 - 数据结构
 - 数据流
 - 数据存储
 - 处理过程

数据字典

- 数据字典是关于数据库中数据的描述，是元数据，而不是数据本身
- 数据字典在需求分析阶段建立，在数据库设计过程中不断修改、充实、完善

7.3 概念结构设计

7.3.1 概念结构

7.3.2 概念结构设计的方法与步骤

7.3.3 数据抽象与局部视图设计

7.3.4 视图的集成

概念结构（续）

- 描述概念模型的工具
 - E-R模型

视图的集成（续）

- 集成局部**E-R**图的步骤

1. 合并

2. 修改与重构

合并分E-R图，生成初步E-R图

- 各分E-R图存在冲突
 - 各个分E-R图之间必定会存在许多不一致的地方
- 合并分E-R图的主要工作与关键
 - 合理消除各分E-R图的冲突

合并分E-R图，生成初步E-R图（续）

- 冲突的种类
 - 属性冲突
 - 命名冲突
 - 结构冲突

7.4 逻辑结构设计

- 逻辑结构设计任务
 - 把概念结构设计阶段设计好的基本E-R图转换为与选用DBMS产品所支持的数据模型相符合的逻辑结构
- 逻辑设计步骤
 - 将概念结构转化为一般的关系、网状、层次模型
 - 将转换来的关系、网状、层次模型向特定DBMS支持下的数据模型转换
 - 对数据模型进行优化

7.4.1 E-R图向关系模型的转换

- 转换内容
- 转换原则

E-R图向关系模型的转换（续）

- E-R图向关系模型的转换要解决的问题
 - 如何将实体型和实体间的联系转换为关系模式
 - 如何确定这些关系模式的属性和码
- 转换内容
 - 将E-R图转换为关系模型：将实体、实体的属性和实体之间的联系转换为关系模式。

E-R图向关系模型的转换（续）

实体型间的联系有以下不同情况：

(1) 一个1:1联系可以转换为一个独立的关系模式，也可以与任意一端对应的关系模式合并。

- 转换为一个独立的关系模式
- 与某一端实体对应的关系模式合并

(2) 一个1:n联系可以转换为一个独立的关系模式，也可以与n端对应的关系模式合并。

- 转换为一个独立的关系模式
- 与n端对应的关系模式合并

E-R图向关系模型的转换（续）

(3) 一个m:n联系转换为一个关系模式。

例，“选修”联系是一个m:n联系，可以将它转换为如下关系模式，其中学号与课程号为关系的组合码：

选修（学号，课程号，成绩）

E-R图向关系模型的转换（续）

(4)三个或三个以上实体间的一个多元联系转换为一个关系模式。

例，“讲授”联系是一个三元联系，可以将它转换为如下关系模式，其中课程号、职工号和书号为关系的组合码：

讲授（课程号，职工号，书号）

E-R图向关系模型的转换（续）

(5)具有相同码的关系模式可合并

- 目的：减少系统中的关系个数
- 合并方法：将其中一个关系模式的全部属性加入到另一个关系模式中，然后去掉其中的同义属性（可能同名也可能不同名），并适当调整属性的次序

10.1 事务的基本概念

一、事务定义

二、事务的特性

一、事务(Transaction)

- 定义
 - 一个数据库操作序列
 - 一个不可分割的工作单位
 - 恢复和并发控制的基本单位
- 事务和程序比较
 - 在关系数据库中，一个事务可以是一条或多条**SQL**语句,也可以包含一个或多个程序。
 - 一个程序通常包含多个事务

二、事务的特性(ACID特性)

事务的ACID特性：

- 原子性（Atomicity）
- 一致性（Consistency）
- 隔离性（Isolation）
- 持续性（Durability）

10.4 恢复的实现技术

- 恢复操作的基本原理：冗余

利用存储在系统其它地方的冗余数据来重建数据库中已被破坏或不正确的那部分数据

- 恢复机制涉及的关键问题

1. 如何建立冗余数据

- 数据转储（**backup**）
- 登录日志文件（**logging**）

2. 如何利用这些冗余数据实施数据库恢复

11.1 并发控制概述

- 并发控制机制的任务
 - 对并发操作进行正确调度
 - 保证事务的隔离性
 - 保证数据库的一致性

并发控制概述（续）

- 并发操作带来的数据不一致性
 - 丢失修改（Lost Update）
 - 不可重复读（Non-repeatable Read）
 - 读“脏”数据（Dirty Read）
- 记号
 - $R(x)$:读数据 x
 - $W(x)$:写数据 x

并发控制概述（续）

- 并发控制的主要技术
 - 有封锁(Locking)
 - 时间戳(Timestamp)
 - 乐观控制法
- 商用的**DBMS**一般都采用封锁方法