南京大学 NANJING UNIVERSITY

奋进行动 团结奋斗 争先进位 坐言起行 应势而动

数 据 管 理 基 础

第6章 关系数据理论

（问题的提出）

智能软件与工程学院

关系数据理论

- 在关系数据库系统的建立过程中，如何设计出一个合适的关系数据库模式?
  - ① 如何评价关系模式设计的好坏?
  - ② 如何设计性能良好的关系模式?

- 关系数据库的规范化理论，就是为了解决上述两个问题而提出的关系数据库设计理论

- 学习目的
  - 了解不好的关系模式设计所存在的问题
  - 理解数据依赖（包括函数依赖和多值依赖）及其相关概念
  - 理解关系规范化理论及其相关算法
  - 关系规范化理论的应用与实践

关系数据理论

关系模式 与 第一范式（1NF）

- 关系模式由五部分组成，是一个五元组：$R(U, D, DOM, F)$
  - ***R***：关系名
  - ***U***：组成该关系的属性名集合
  - ***D***：***U*** 中属性所来自的域
  - **DOM**：属性到域的映射集合（描述各个属性对应的域）
  - ***F***：属性组***U***上的一组数据依赖（关系上的完整性约束条件）
  - 关系名***R***是符号化的元组语义

- 关系的性质
  - ① 列是同质的
    - 每一列的分量来自同一个域
  - ② 不同的列可出自同一个域
  - ③ 列的无序性（属性的无序性）
  - ④ 行的唯一性（元组的唯一性）
  - ⑤ 行的无序性（元组的无序性）
  - ⑥ 分量必须取原子值
    - 每个分量都是不可分的数据项

- 由于***D***、**DOM**与模式设计关系不大，因此可把关系模式看作一个三元组 $R<U,F>$
  - 作为关系的性质：每个分量都是不可分的数据项
  - 满足了上述六条性质的关系模式就属于**第一范式（1NF）**

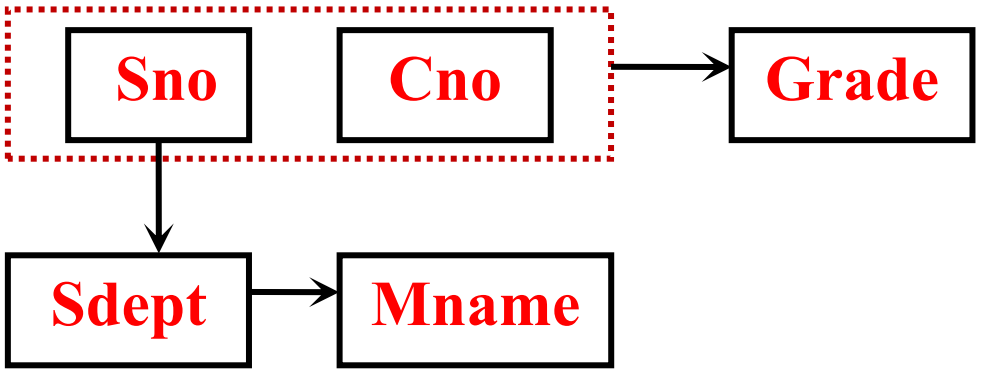
关系规范化理论

- [例6.1] 建立一个描述学校教务的数据库。涉及的对象包括：
  - 学生的学号 Sno、所在系 Sdept
  - 系主任姓名 Mname
  - 课程号 Cno
  - 成绩 Grade

- 现实世界的已知事实（语义）：
  - 一个系有若干学生，但一个学生只属于一个系；
  - 一个系只有一名（正职）系主任，允许一个人兼任多个系的系主任；
  - 一个学生可以选修多门课程，每门课程有若干学生选修；
  - 一个学生学习一门课程仅有一个成绩。

- 由此可得到属性组U上的一组函数依赖F：

$$F = \{\ Sno \rightarrow Sdept,\ Sdept \rightarrow Mname,\ (Sno, Cno) \rightarrow Grade\ \}$$



关系规范化理论

- 关系模式设计
  - 同一个关系数据库系统可以有多种不同的关系模式设计方案
- 不同模式设计方案的比较
  - 不同的模式设计方案有好坏之分
  - 好的设计方案应该是：既具有合理的数据冗余度，又没有插入和删除等操作异常现象
- 在不同的设计结果之间产生区别的原因
  - 在一个关系中，属性与属性之间存在语义相关性（数据依赖）
  - 不同的设计方案，可能会产生不同的数据依赖
- 关系的规范化
  - 按照给定范式的要求来设计关系模式
  - 范式(Normal Form)：对一个关系中允许存在的函数依赖的要求

- 同一个数据库系统可以有多种不同的模式设计方案。
  - [例6.1] 假设一个学生数据库中有五个属性：学号*Sno*，就读院系*Sdept*，系主任姓名*Mname*，课程号*Cno*，课程成绩*Grade*，可以采用的模式设计方案有多种（如下所示）

方案1：用一个单一的关系模式来表示

$Student(Sno, Sdept, Mname, Cno, Grade)$

函数依赖为：$\{Sno \rightarrow Sdept, Sdept \rightarrow Mname, (Sno, Cno) \rightarrow Grade\}$

方案2：用三个关系模式来表示

- $S(Sno, Sdept)$ 函数依赖为 $Sno \rightarrow Sdept$
- $DEPT(Sdept, Mname)$ 函数依赖为 $Sdept \rightarrow Mname$
- $SC(Sno, Cno, Grade)$ 函数依赖为 $(Sno, Cno) \rightarrow Grade$

关系规范化理论 - 不同模式设计方案的比较 1

❑方案一（单个关系Student）

Sno	Sdept	Mname	Cno	Grade
S1	计算机	张明	C1	95
S2	计算机	张明	C1	90
S3	计算机	张明	C1	88
S4	计算机	张明	C1	70
S5	计算机	张明	C1	78
S1	计算机	张明	C2	86
S3	计算机	张明	C2	94
S6	数学	李刚	C2	92
S7	数学	李刚	C3	85
S8	人工智能	张明	C3	87

❑方案二（多个关系）

S

Sno	Sdept
S1	计算机
S2	计算机
S3	计算机
S4	计算机
S5	计算机
S6	数学
S7	数学
S8	人工智能

DEPT

Sdept	Mname
计算机	张明
数学	李刚
人工智能	张明

SC

Sno	Cno	Grade
S1	C1	95
S2	C1	90
S3	C1	88
S4	C1	70
S5	C1	78
S1	C2	86
S3	C2	94
S6	C2	92
S7	C3	85
S8	C3	87

关系规范化理论 - 不同模式设计方案的比较 2

- 下面从三个方面来比较这两种数据模型设计方案：

	方案1 （单个关系）	方案2 （三个关系）
数据冗余	高	低
更新异常	有	无
插入异常	有	无
删除异常	有	无

- 在一个不好的关系模式设计方案中（方案1），会存在数据冗余度大，以及因此带来的元组更新异常、插入异常和删除异常现象。

(1) redundency（数据冗余）

- 浪费大量的存储空间
- 例如：每一个系主任的姓名，会重复出现在该系的每一位同学的每一门选课元组中。

Sno	Sdept	Mname	Cno	Grade
S1	计算机	张明	C1	95
S2	计算机	张明	C1	90
S3	计算机	张明	C1	88
S4	计算机	张明	C1	70
S5	计算机	张明	C1	78
S1	计算机	张明	C2	86
S3	计算机	张明	C2	94
S6	数学	李刚	C2	92
S7	数学	李刚	C3	85
S8	人工智能	张明	C3	87

(2) abnormity of update（更新异常）

- 因为数据的冗余存储，导致update操作被重复执行，既增加了执行上的时间开销，也可能因为修改不彻底而面临数据不一致的危险。
- 例如：某系更换系主任后，必须修改与该系学生有关的每一个元组。

Sno	Sdept	Mname	Cno	Grade
S1	计算机	张明	C1	95
S2	计算机	张明	C1	90
S3	计算机	张明	C1	88
S4	计算机	张明	C1	70
S5	计算机	张明	C1	78
S1	计算机	张明	C2	86
S3	计算机	张明	C2	94
S6	数学	李刚	C2	92
S7	数学	李刚	C3	85
S8	人工智能	张明	C3	87

(3) abnormity of insert（插入异常）

- 可能因为违反‘实体完整性’约束而导致元组插入失败！
- 例如：如果一个系刚成立，尚无学生，则无法把这个系（信息安全）及其系主任（王必成）的信息存入数据库。

Sno	*Sdept*	*Mname*	*Cno*	*Grade*
S1	计算机	张明	C1	95
…	…	…	…	…
S6	数学	李刚	C2	92
…	…	…	…	…
S8	人工智能	张明	C3	87
	信息安全	王必成		

X

- 原因：该关系的候选码由**Sno**和**Cno**联合构成。如果允许插入一条没有学生和课程信息的元组（信息安全，王必成），显然违反了关系上的实体完整性约束，该条元组插入操作必然执行失败！

(4) abnormity of delete（删除异常）

- 执行元组删除操作，可能连带删除掉一些本不该被删除的信息！

- 例如：如果某个系的学生全部毕业了，则在删除该系学生信息的同时，把这个系及其系主任的信息也丢掉了。

- 如右图所示，如果删除学号为S8的学生（人工智能学院唯一的一位同学），数据库中就不再有人工智能学院的信息！

Sno	Sdept	Mname	Cno	Grade
S1	计算机	张明	C1	95
S2	计算机	张明	C1	90
S3	计算机	张明	C1	88
S4	计算机	张明	C1	70
S5	计算机	张明	C1	78
S1	计算机	张明	C2	86
S3	计算机	张明	C2	94
S6	数学	李刚	C2	92
S7	数学	李刚	C3	85

(删除学生S8后的关系Student)

关系规范化理论 - 不同模式设计方案的比较 3

- 关系模式Student<U, F>中存在的问题（总结）
  - 数据冗余度高
    - 浪费大量的存储空间
  - 更新异常（Update Anomalies）
    - 因为数据的冗余存储，导致update操作被重复执行，既增加了执行上的时间开销，也可能因为修改不彻底而面临数据不一致的危险。
  - 插入异常（Insertion Anomalies）
    - 可能因为违反‘实体完整性’约束而导致元组插入失败。
  - 删除异常（Deletion Anomalies）
    - 可能连带删除掉一些本不该被删除的信息。

关系规范化理论 - 不同模式设计方案的比较 4

- 结论
  - Student关系模式不是一个好的模式。
  - 一个“好”的模式应当不会发生插入异常、删除异常和更新异常，数据冗余应尽可能少。

- 原因
  - 由存在于模式中的某些数据依赖引起的。

- 解决方法
  - 用规范化理论改造关系模式来消除其中不合适的数据依赖
  - 实现方法：模式分解（将不好的关系模式，分解成为若干个子模式）

关系规范化理论 - 不同模式设计方案的比较 5

- 如果采用方案二，这三个模式都不会发生插入异常、删除异常和更新异常的问题，数据的冗余也得到了控制。

- 数据冗余：系主任的姓名不会被重复存储

- 更新异常：更换系主任，只需要在关系DEPT中修改对应系的Mname值

- 插入异常：只需要把新成立的系（信息安全，王必成）插入到DEPT表中

- 删除异常：欲删除学生S8，只需要在学生关系S和选课关系SC中删除相关元组，他就读的院系(人工智能,张明)仍然保留在数据库中。

S

Sno	Sdept
S1	计算机
S2	计算机
S3	计算机
S4	计算机
S5	计算机
S6	数学
S7	数学
S8	人工智能

DEPT

Sdept	Mname
计算机	张明
数学	李刚
人工智能	张明

SC

Sno	Cno	Grade
S1	C1	95
S2	C1	90
S3	C1	88
S4	C1	70
S5	C1	78
S1	C2	86
S3	C2	94
S6	C2	92
S7	C3	85
S8	C3	87

关系规范化理论 - 在不同的设计结果之间产生区别的原因 1

- 原因
  - 在一个关系中，属性与属性之间存在语义相关性（数据依赖）
  - 不同的设计方案，可能会产生不同的数据依赖

- 数据依赖
  - 是一个关系内部属性与属性之间的一种约束关系
    - 通过属性间值的相等与否体现出来的数据间相关联系
  - 是现实世界属性间相互联系的抽象
  - 是数据内在的性质
  - 是语义的体现

- 数据依赖的主要类型
  - 函数依赖（Functional Dependency，简记为FD）
  - 多值依赖（Multi-Valued Dependency，简记为MVD）

关系规范化理论 - 在不同的设计结果之间产生区别的原因 2

- ‘函数依赖’普遍存在于现实生活中
  - 描述一个‘学生’关系，可以有学号、姓名、系名等属性。
    - 一个学号只对应一个学生；
    - 一个学生只有一个名字，且只在一个系中学习；
    - “学号”值确定后，学生的姓名及所在系的值就被唯一确定。

- 根据以上的语义描述，在‘学生’关系中，在属性之间存在类似于数学中的‘函数’的依赖关系：
  - 以Sno为自变量、以Sname为因变量的函数：$Sname = f(Sno)$
    - 即：Sname是Sno的函数，被称为“Sno函数决定Sname”
    - 记作 $Sno \rightarrow Sname$
  - 以Sno为自变量、以Sdept为因变量的函数：$Sdept = f(Sno)$
    - 即：Sdept是Sno的函数，被称为“Sno函数决定Sdept”
    - 记作 $Sno \rightarrow Sdept$

关系规范化理论 - 在不同的设计结果之间产生区别的原因 3

- [例6.1] 对比这两种不同的模式设计方案，虽然看上去函数依赖没有什么变化，但具体到每一个关系，函数依赖还是有了不同。

方案1：用一个单一的关系模式来表示

$Student(Sno, Sdept, Mname, Cno, Grade)$

函数依赖为：$\{Sno \rightarrow Sdept, Sdept \rightarrow Mname, (Sno, Cno) \rightarrow Grade\}$

方案2：用三个关系模式来表示

- $S(Sno, Sdept)$ 函数依赖为 $Sno \rightarrow Sdept$
- $DEPT(Sdept, Mname)$ 函数依赖为 $Sdept \rightarrow Mname$
- $SC(Sno, Cno, Grade)$ 函数依赖为 $(Sno, Cno) \rightarrow Grade$

  - 要设计出性能良好的关系模式，必须从数据库中所有属性的语义上进行分析，分清每个属性的语义含义及其相互之间的依存关系。进而将那些相互依赖密切的属性组合在一起构成一个关系模式，避免对属性的松散组合所引起的‘排它性’，从而可以降低数据冗余度，避免上述异常现象的产生。

关系规范化理论 - 关系的规范化 1

- 不好的模式设计方案，是因为关系中的数据依赖存在某些不好的性质（或者说，存在着某些不好的数据依赖）

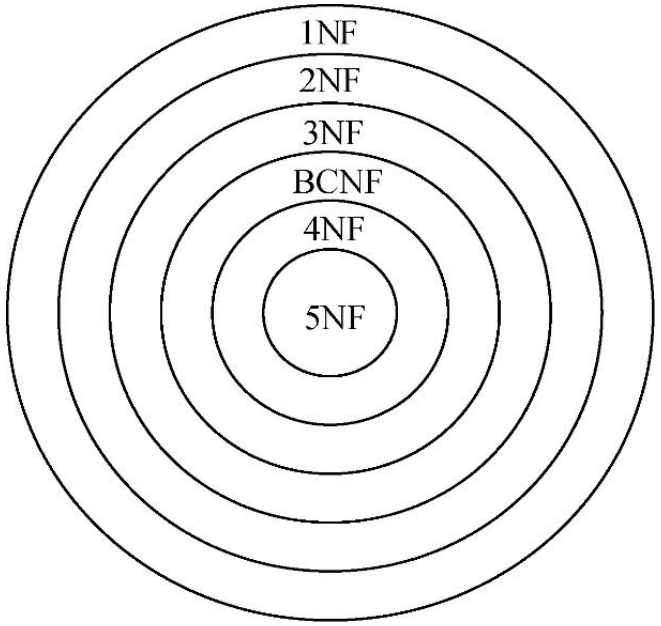
- 范式 (Normal Form)
  - 是对一个关系中允许存在的数据依赖的要求
  - 根据对数据依赖要求程度的不同，又分为不同级别的‘范式’
    - 从低到高依次分为：1NF, 2NF, 3NF, BCNF, 4NF, 5NF
  - 范式也可被理解为是符合某一种级别的关系模式的集合

- 关系规范化设计
  - 按照给定范式的要求来设计关系模式，被称为‘关系规范化设计’
  - 相关理论被称为‘关系规范化理论’

关系规范化理论 - 关系的规范化 2

❑ 各种范式之间的联系

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



- 如果一个关系模式满足高级别范式对数据依赖的要求，那么它肯定也满足低级别范式对函数依赖的要求；
- 一个关系模式满足低级别范式对数据依赖的要求，但它不一定能满足高级别范式对函数依赖的要求。
- 如果某一关系模式R满足范式n的要求，则称关系模式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（schema decomposition）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（normalization）。

- 将分以下五个主题来介绍本章的内容
  - 函数依赖 与 码
  - 范式 与 规范化
  - Armstrong公理系统
  - 模式的分解
  - 多值依赖 与 4NF