第10章

数据依赖与关系模式规范化

2012.04

目录 Contents

10.1 概述

- 10.1.1 关系模式的设计
- 10.1.2 “不好的”（Bad）关系模式
- 10.1.3 如何设计“好的”关系模式
- 10.1.4 权衡：规范化 & 性能

10.1.1 关系模式的设计

- 概念回顾
  - 关系
    - 描述实体、属性、实体间的联系。
    - 从形式上看，它是一张二维表，是所涉及属性的笛卡儿乘积的一个子集。
  - 关系模式
    - 运用关系数据模型对一个企业/组织的一组数据的结构、联系和约束进行描述的结果。
    - 实际上是用来定义关系的。
  - 关系数据库
    - 基于关系模型的数据库，利用关系来描述现实世界。
    - 从形式上看，它由一组关系组成。

10.1.1 关系模式的设计

- 关系模式的设计：成功开发数据库应用系统的关键
  - DB主要用于支持数据密集型应用 (Data Intensive Applications)
  - 数据密集型应用的核心问题是: DB设计
  - DB设计－结构方面：
    - 数据模式(Data Schemas), e.g. a Set of Relational Schemas
      - 面向过程的方法(Process-oriented), e.g. SA/SD
      - 面向数据的方法(Data-oriented), e.g. IEM √
    - 数据模型(Data Model), e.g. Relational Model
  - 目标：设计一个“好的”(Good)关系模式－关系数据库的逻辑设计。
    - But, What is a good relational schema?
    - 数据库逻辑设计的工具－关系数据库的规范化理论

10.1.2 “不好的”（Bad）关系模式

- 1. 模式设计
  - 同一个数据库系统可以有多种不同的模式设计方案
  - 如假设一个学生数据库中有以下属性：学号(SNO), 课程号(CNO), 成绩(G), 任课教师姓名(T), 教师所在系名(DEPT)。这些数据具有下列语义：
    - 学号是一个学生的标识，课程号是一门课程的标识，这些标识与其表的学生和课程分别一一对应；
    - 一位学生所修的每门课程都有一个成绩；
    - 每门课程只有一位任课教师，但一位教师可以教多门课程；
    - 教师中没有重名，每位教师只属于一个系。
  - 可以采用的模式设计方案有多个

10.1.2 “不好的” 关系模式

- 方案一
  - 一个关系
    - R (SNO, CNO, G, T, DEPT)
- 方案二
  - 三个关系
    - R1 (SNO, CNO, G)
    - R2 (CNO, T)
    - R3 (T, DEPT)

10.1.2 “不好的”关系模式

- 方案一
  - 对于关系模式R(SNO, CNO, G, T, DEPT)的一个实例

R

SNO	CNO	G	T	DEPT
95601	C01	85	张乐	计算机
95602	C01	90	张乐	计算机
95801	C01	95	张乐	计算机
95801	M03	91	王丽	数理
95802	M03	88	王丽	数理

Table1

10.1.2 “不好的”关系模式

- 方案二
  - 对于关系模式R1 (SNO, CNO, G)，R2 (CNO, T)，R3 (T, DEPT)的一个实例

SNO	CNO	G
95601	C01	85
95602	C01	90
95801	C01	95
95801	M03	91
95802	M03	88

R1

CNO	T
C01	张乐
M03	王丽

R2

T	DEPT
张乐	计算机
王丽	数理

R3

Table2

10.1.2 “不好的”关系模式

- 2 不同模式设计方案的比较
  - 不同的模式设计方案对数据库的影响是否相同？
  - 例：根据方案1所建立的数据库如表1所示，根据方案2所建立的数据库如表2所示。
  - 我们从下面几个方面来比较这两个数据库：
    - 数据冗余度
    - 元组插入操作
    - 元组删除操作
    - 元组更新操作

10.1.2 “不好的”关系模式

- 经过比较发现，表1具有如下问题：
  - 冗余(Redundancy)
    - 重复多次：“C01”课的教师是“张乐”；“张乐”是“计算机”系的教师。
    - 对于方案二，则不存在数据冗余
  - 异常(Anomalies)
    - 更新异常(Update Anomalies)
      - “张乐”调到“土木”系，而只改了其中一个元组的值，出现数据不一致。
      - “M03”课的教师换成“杨萍”， 而只改了其中一个元组的值，出现数据不一致。
      - 对于方案二，只需分别修改R2和R3关系中的元组的值即可，不会出现数据不一致。

10.1.2 “不好的”关系模式

- 删除异常(Delete Anomalies)
  - “C01”课不开了，需删除表1中的前三个元组，“张乐”是“计算机”系的教师的信息也随着被删除。
  - 对于方案二，我们可以仅在关系R1和R2关系中分别删除课程为“C01”的元组信息，但不会误删除掉“张乐”是“计算机”系教师的信息，其所对应的元组仍然保留在关系R3中。

- 插入异常(Insert Anomalies)
  - 如果需要新开设一门尚未有学生选修的课程(C05, 许卓明)，则无法构造出一个由SNO, CNO, G, T, DEPT属性值所组成的新元组，在表1中就无法执行元组的插入操作。
  - 对于方案二，我们可以直接将元组(C05,许卓明)插入到关系R2中。

10.1.2 “不好的”关系模式

- 因此，不同的模式设计方案有好坏的区分。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异常现象的出现。

- 3 在不同的设计结果之间产生区别的原因
  - 数据库的各属性之间是互相关联的，它们互相依赖、互相制约，构成一个结构严密的整体。
  - 要设计出一个好的关系模式，必须从数据库中所有属性的语义上进行分析，从语义上入手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。

10.1.2 “不好的”关系模式

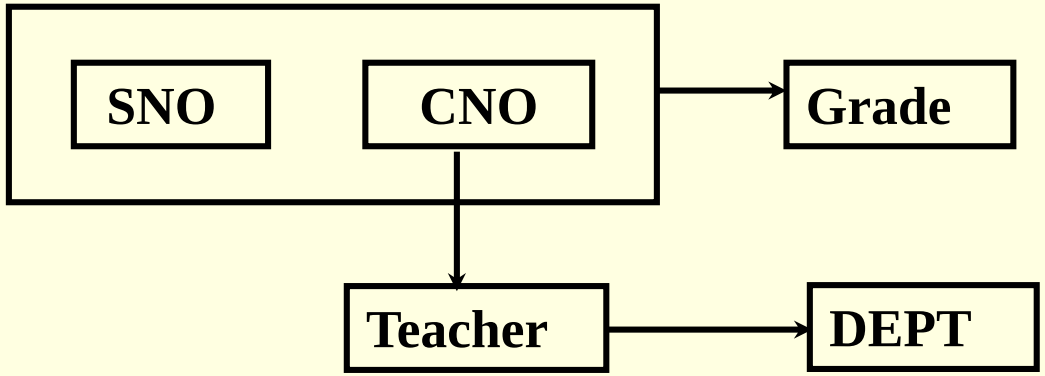
- 关系模式中数据的“语义”不单纯。在此，“语义”专指问题空间中固有的、相对稳定的数据依赖(DD)关系。
- 数据依赖
  - 是通过一个关系中属性间值的相等与否体现出来的数据间的相互关系，是现实世界属性间相互联系的抽象，是语义的体现。
  - e.g. 函数依赖（Functional Dependency，FD）
    - 一个/组属性X的值是否决定另一个/组属性Y的值。$X \xrightarrow{?} Y$
  - 多值函数依赖（Multivalued Dependency，MVD）
  - 连接依赖（Join Dependency，JD）

10.1.2 “不好的”关系模式

- 由于在关系模式R中存在某些数据依赖，引起数据冗余和数据更新异常。解决方法：通过分解关系模式来消除其中不合适的数据依赖，即关系规范化。
- 对以上模式R(SNO, CNO, G, T, DEPT), 有以下三个函数依赖：
  - 1. SNO,CNO→G　　2. CNO→T　　3. T→DEPT

相应的表示了三个事实，为何不用三个模式呢？

R1(SNO,CNO,G)　2. R2(CNO,T)　3. R3(T,DEPT)



10.1.3 如何设计“好的”关系模式

- 如何设计“好的”关系模式：规范化, 模式分解 & 范式
  - 规范化(Normalization)
    - 将一个关系模式按“语义单纯化”的原则进行合理的分解----称模式分解(Decomposition)，以最终达到“一事一地(One Fact in One Place)”。
      - 一个关系中
      - 一个实体/联系
    - 在每个关系中，属性与属性之间一定要满足某种内在的语义联系，这被称为关系的规范化。
  - 模式分解的条件 / 准则
    - 起码：分解是无损的(Lossless)：分解前后要等价，即对任何相同的查询总是产生相同的结果。(可通过“连接”分解后的诸关系重构原关系)。
    - 理想：分解是保持依赖的(Preserving Dependencies)：这需进一步论述。

10.1.3 如何设计“好的”关系模式

- 范式(Normal Form)
  - 规范化(即模式分解)程度的一种测度。根据对属性间所存在的内在语义联系要求的不同，又可以将关系的规范化分为若干个级别，这被称为范式。

低

第一范式(1NF)

第二范式(2NF)　　函数依赖(FD)范畴 √

第三范式(3NF)

Boyce-Codd 范式(BCNF)

第四范式(4NF)　　多值函数依赖(MVD)范畴

第五范式(5NF)　　连接依赖(JD)范畴

高

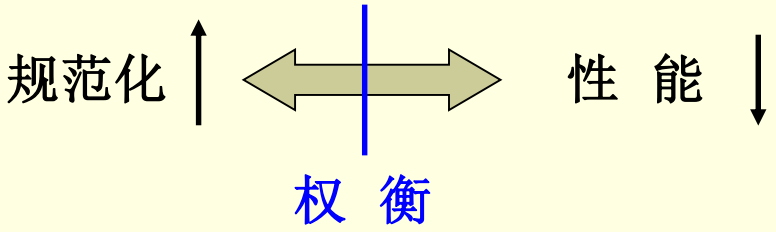
  - 一个关系模式R达到x范式的程度称：R is in xNF，记为：$R \in xNF$；否则，称：R violates xNF condition, 记为：$R \notin xNF$。

10.1.3 如何设计“好的”关系模式

- 数据依赖理论
  - 函数依赖（FD – Functional Dependency）
    - 1970年，E. F. Codd
    - 1972 – 1974年，Codd, Casey, Bernstein, Armstrong
      - 完全/部分FD，平凡/非平凡FD，直接/传递FD
      - 键（key）
    - 1974年，Armstrong公理系统
      - FD的逻辑蕴涵
      - FD的形式化推理规则集
  - 多值依赖（MVD – Multi-Valued Dependency）
    - 1976 – 1978年，Zaniolo, Fagin, Delobel

10.1.4 权衡：规范化 & 性能

- 规范化程度并非越高越好



- 程度
  - 一般到BCNF/3NF已足够

- 策略
  - 对更新频繁/查询较少的表：规范化↑ ------减少“异常”
  - 对查询频繁/更新较少的表：规范化↓ ------提高“性能”

目录 Contents

10.2 函数依赖与范式

10.2.1 函数依赖

- 函数依赖
  - 完全/部分FD，平凡/非平凡FD，直接/传递FD
- Armstrong公理系统
- 键（key）
- 两个算法：
  - 属性集的闭包计算
  - 关键字的计算

10.2.2 与函数依赖有关的范式

- 范式：1NF，2NF，3NF，BCNF

10.2.1 函数依赖

- 在一个关系模式R(U)中，如果一部分属性Y的取值依赖于另一部分属性X的取值，则在属性集X和属性集Y之间存在着一种数据依赖关系，我们称之为函数依赖。

  - 例1：在学生关系模式 Student(SNO, Sname, Sdept, Sage) 中就存在下面的几组依赖关系：
    - { Sname } 的取值依赖于 { SNO } 的取值
    - { Sdept } 的取值依赖于 {SNO} 的取值
    - { Sage } 的取值依赖于 {SNO} 的取值

  - 例2：在选课关系模式SC(SNO, CNO, Grade)中：
    - { Grade } 的取值依赖于 { SNO, CNO } 的取值

10.2.1 函数依赖

- 定义：函数依赖 / 决定子(Determinant)
  - 设有关系模式R，其中U是属性集，A、B是U的子集。若对于关系模式R(U)的任一关系实例 r 中的任两个元组 t 和s，

    t [A] = s [A] $\Longrightarrow$ t [B] = s [B] 成立，
  - 则称B函数依赖于A或A函数决定B，记为：A → B。A称为决定子。
  - 亦可记为：A1, A2, … , An → B1, B2, … , Bm (Ai，Bj为单个属性)。
  - 注：A不函数决定B，记为A↛B。若A → B，B → A，则称一一对应，记为A$\longleftrightarrow$B。

10.2.1 函数依赖

- 分裂/合并规则(The Splitting/Combining Rule)

X1, X2, … , Xn → Y1, Y2, … , Ym ⟺ X1, X2, … , Xn → Y1

…….

X1, X2, … , Xn → Ym

10.2.1 函数依赖

- 函数依赖是语义范畴上的概念，只有根据属性间固有的语义联系才能归纳出与客观事实相符合的函数依赖关系，而不是仅从现有的一个或若干个关系实例中得出的结论。
- 特定的关系实例虽然不能用于函数依赖的发现，但可以用于否定某些函数依赖。

SNO	Sname	Sdept	Sage
0001	王剑飞	CS	17
0002	陈　瑛	MA	19
0003	方世觉	CS	17

关系 Student

10.2.1 函数依赖

- 函数依赖反映的是同一个关系中的两个属性子集之间在取值上的依存关系，这种依存关系实际上也是一种数据完整性约束。因此，我们也可以通过对数据完整性约束条件的分析来寻找属性之间的函数依赖关系。
- 一个学生数据库中有以下属性：学号(SNO), 课程号(CNO), 成绩(G), 任课教师姓名(T), 教师所在系名(DEPT)。这些数据具有下列语义：
  - 学号是一个学生的标识，课程号是一门课程的标识，这些标识与其表的学生和课程分别一一对应；
  - 一位学生所修的每门课程都有一个成绩；
  - 每门课程只有一位任课教师，但一位教师可以教多门课程；
  - 教师中没有重名，每位教师只属于一个系。

  - 有以下三个函数依赖：

    1. SNO,CNO→G　　2. CNO→T　　3. T→DEPT

10.2.1 函数依赖

- 定义：平凡依赖 / 非平凡依赖 / 完全非平凡依赖

  设A, B是某模式的两个属性集，对函数依赖A→B，

  - 若B⊆A，则称此函数依赖为平凡依赖(Trivial Dependency);
  - 若B – A≠Φ，则称此函数依赖为非平凡依赖(Nontrivial Dependency);
  - 若B∩A = Φ，则称此函数依赖为完全非平凡依赖(Completely Nontrivial Dependency)。

  - 例：在关系SC (Sno, Cno, Grade)中，

    非平凡函数依赖： (Sno, Cno) → Grade

    平凡函数依赖： (Sno, Cno) → Sno

    (Sno, Cno) → Cno

  - 对于任一关系模式，平凡函数依赖都是必然成立的，它不反映新的语义，因此若不特别声明，我们总是讨论非平凡函数依赖。

10.2.1 函数依赖

- 平凡依赖规则(The Trivial-dependency Rule)
  - A →B ⟺A→B–A。

- 定义: 完全依赖 / 部分依赖
  - 设A, B是某模式的两个不同属性集, 若有A→B，且不存在C⊂A，使C →B，则称A →B为完全依赖 (Full Dependency), 记为：$A \xrightarrow{f} B$；否则称为部分依赖(Partial Dependency), 记为：$A \xrightarrow{p} B$。

- 例: 在关系SC (Sno, Cno, Grade)中，由于：Sno ↛Grade，Cno ↛ Grade，　因此：(Sno, Cno) $\xrightarrow{f}$ Grade。

10.2.1 函数依赖

- 定义：传递依赖
  - A, B, C是某模式的三个不同属性集，若有：A→B, B↛A, B→C, 则称C传递函数依赖于A, 记为：$A \xrightarrow{t} C$。
  - 为了使得函数依赖在表示形式上的简单化，传递函数依赖与非传递函数依赖在表示形式上没有区别。

- 传递规则(The Transitive Rule)
  - A→B , B→C ⇨ A→C。

- 例: 在关系Std (Sno, Sdept, Deanname)中，有：

  Sno → Sdept，Sdept → Deanname，所以, Deanname传递函数依赖于Sno，即Sno→Deanname

10.2.2 范式

- 回顾概念
  - 键(key)：在关系模式R(U)中，如有K⊆U且满足：$K \xrightarrow{f} U$，则K称为R的一个侯选键(Candidate Key)，简称键。也就是说：
    - 决定性：这个属性（组）K的值唯一地决定了其他属性的值（因而也决定了整个元组）；
    - 最小性条件：这个属性（组）K的任何真子集均不满足决定性条件。
  - 主键(Primary Key)：在关系模式机器实现时，若关系模式R有多个候选码，则选定其中的一个做为主键。其他键称为候补键(Alternate Key)。
  - 超键(Superkey)：关系模式中包含键的属性(组)称为超键。
  - 全键(all key)：主键是由所有的属性构成的，这称为全键。

10.2.2 范式

- 回顾概念(cont.)
  - 主属性(集)
    - 由关系模式 R 的所有键中的属性所构成的集合被称为关系模式 R 的 主属性集。
    - 主属性集中的属性被称为关系模式 R 的 主属性。
  - 非主属性(集)
    - 由主属性集之外的其它属性所构成的集合被称为关系模式 R 的 非主属性集。
    - 非主属性集中的属性被称为关系模式 R 的 非主属性。

10.2.2 范式

- 范式(Normal Form)
  - 规范化(即模式分解)程度的一种测度。根据对属性间所存在的内在语义联系要求的不同，又可以将关系的规范化分为若干个级别，这被称为范式。

低 ↓ 高

第一范式(1NF)	
第二范式(2NF)	函数依赖(FD)范畴 √
第三范式(3NF)	
Boyce-Codd 范式(BCNF)	
第四范式(4NF)	多值函数依赖(MVD)范畴
第五范式(5NF)	连接依赖(JD)范畴

  - 一个关系模式R达到x范式的程度称：R is in xNF，记为：$R \in xNF$；否则，称：R violates xNF condition, 记为：$R \notin xNF$。

10.2.2 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。
- 各种范式之间存在联系：

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

10.2.2 范式

- 定义: 1NF
  - 设有一个关系模式R, 若R的任一关系实例r中的属性值均是原子数据(属性都是不可分的基本数据项), 则称R属于1NF，记为R∈1NF。
  - 注
    - 1NF条件是传统关系数据库系统的基本要求，目前大多数RDBMS均要求如此。
    - 但是满足第一范式的关系模式并不一定是一个好的关系模式。
    - 突破这一条件称非第一范式(non-first normal form, NF^2)条件。
    - E/R中的非原子属性在转化为关系时要这样处理：
      - 对集合属性：纵向展开
      - 对元组属性：横向展开

10.2.2 范式

- 定义: 2NF
  - 设有一个关系模式R∈1NF, 若R的每个非主属性均完全函数依赖于键, 则称R属于2NF，记为R∈2NF。
  - 注
    - **A.** R∈2NF $\Longrightarrow$ R∈1NF
    - **B.** 考察 R(SNO, CNO, G, T, DEPT)∈1NF,

      因 SNO, CNO→G,

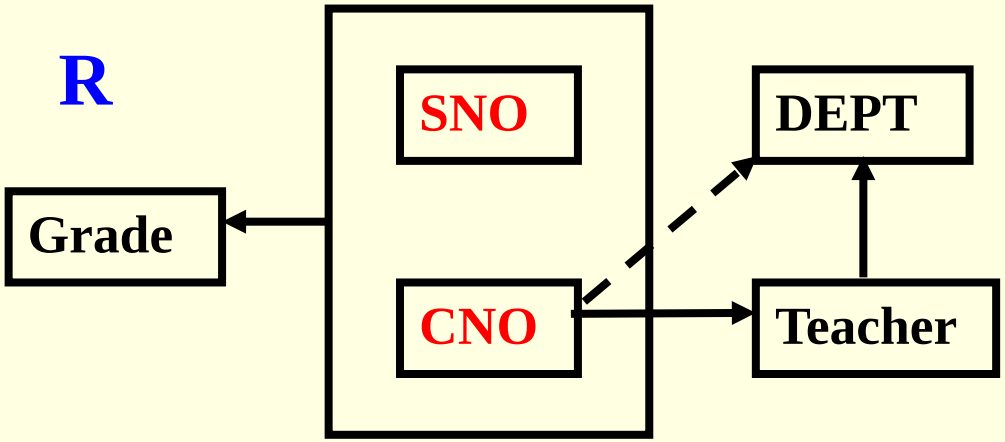
      CNO→T, T→DEPT $\Longrightarrow$ CNO→DEPT $\Longrightarrow$ CNO→T, DEPT

      故Key={SNO, CNO}。但Key $\xrightarrow{P}$ T, DEPT,

      故R$\notin$2NF 。

10.2.2 范式

- R(SNO, CNO, G, T, DEPT)不是一个好的关系模式
  - 数据冗余度大
  - 插入异常
  - 删除异常
  - 更新复杂
- 原因
  - T, DEPT部分函数依赖于键{SNO, CNO}。



10.2.2 范式

- 解决方法
  - 模式分解：消除这些部分函数依赖

    R1(SNO, CNO, G) ∈2NF，R2(CNO, T, DEPT)∈2NF。
    - 但R2仍有问题：

      c01 t1 d1

      c02 t1 d1　　冗余

      c03 t1 d1

      c04 t2 d2
    - 原因：Key={CNO}，因T→DEPT , 而T非超键, DEPT又非主属性。
- 采用投影分解法将一个1NF的关系分解为多个2NF的关系，可以在一定程度上减轻原1NF关系中存在的插入异常、删除异常、数据冗余度大、修改复杂等问题。
- 将一个1NF关系分解为多个2NF的关系，并不能完全消除关系模式中的各种异常情况和数据冗余。

10.2.2 范式

- 定义: 3NF
  - 设有一个关系模式R∈1NF，若R的任一非平凡函数依赖X→A满足下列两个条件之一: (1) X是超键, (2)A是主属性，则称R属于3NF，记为R∈3NF。
  - 注
    - **A.** 若R∉3NF，意味着：A非主属性，X又非超键：
      - (1) X是键的真子集：非主属性A部分依赖于键；
      - (2) X既非超键又非键的真子集：

      的部分依赖和传递依赖。

      B. R∈3NF　　R∈2NF。Key →X, X→A　　Key→A，即：非主属性A传递依赖于键。

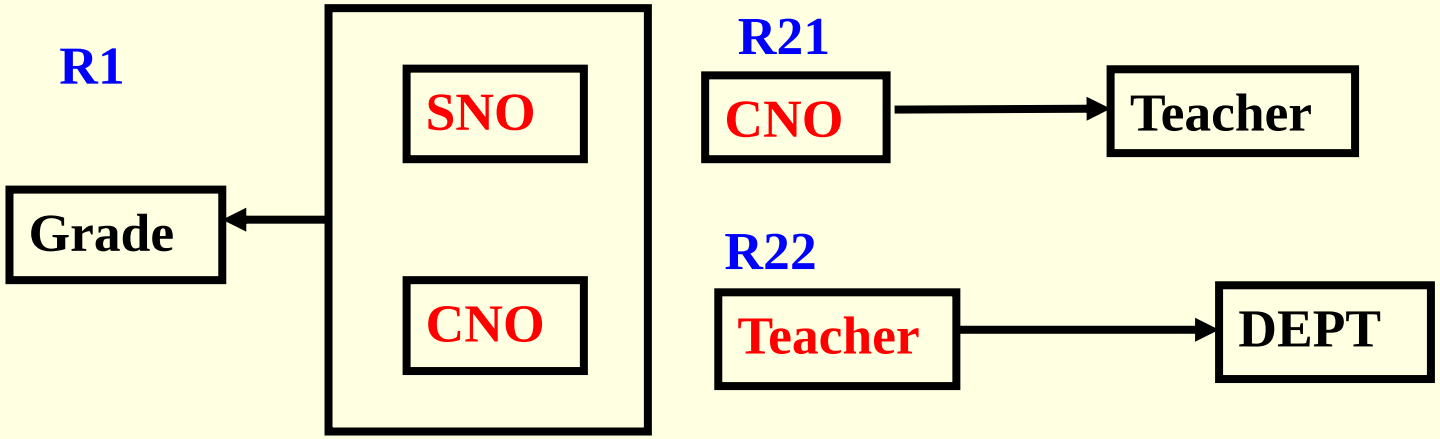
      故3NF消除了非主属性对键

10.2.2 范式

- 例：将关系模式R(SNO, CNO, G, T, DEPT)分解得到

  R1(SNO, CNO, G) ∈2NF，R2(CNO, T, DEPT)∈2NF，

  由于关系模式R2中存在非主属性DEPT对键的传递依赖，所以采用投影分解法，把R2分解为两个关系模式，以消除传递函数依赖：R21(CNO, T) ∈3NF, R22(T, DEPT) ∈ 3NF，即CNO →T, T → DEPT。



10.2.2 范式

- 若R∈3NF，则R的每一个非主属性既不部分函数依赖于候选键也不传递函数依赖于键。
- 采用投影分解法将一个2NF的关系分解为多个3NF的关系，可以在一定程度上解决原2NF关系中存在的插入异常、删除异常、数据冗余度大、修改复杂等问题。
- 将一个2NF关系分解为多个3NF的关系后，并不能完全消除关系模式中的各种异常情况和数据冗余。

10.2.2 范式

- 定义: BCNF
  - 设有一个关系模式R∈1NF，若R的任一非平凡函数依赖X→A满足下列条件：决定子X必是超键，则称R属于BCNF，记为R∈BCNF。
  - 注：
    - **A.** R∈BCNF ⟹ R∈3NF。
    - **B.** BCNF消除了(非主/主)属性对键的部分依赖和传递依赖。
    - **C.** 关系模式达到BCNF后，在函数依赖范畴内已彻底规范化了，并消除了冗余和异常。
    - **D. 任何全键关系模式必属于BCNF**
    - **E. 任何两属性关系模式必属于BCNF**
    - **F. 关系模式无损分解成BCNF的策略**

10.2.2 范式

- D. 任何全键关系模式必属于BCNF
  - 证明：设$R(A_1, A_2, \dots, A_n)$是全键关系模式, 即$Key=\{ A_1, A_2, \dots, A_n \}$,

    反证法：

    假设R∉BCNF，则必存在一非平凡函数依赖$X \rightarrow A_i$, 而决定子X不是超键。

    注意, $X \subset \{ A_1, A_2, \dots, A_n \}$， $A_i \notin X$；

    $X \subset \{ A_1, A_2, \dots, A_n \} - \{A_i\} = \{ A_1, A_2, \dots, A_{i-1}, A_{i+1}, \dots, A_n \}$，故$Key=\{ A_1, A_2, \dots, A_{i-1}, A_{i+1}, \dots, A_n \}$。

    这与全键的条件矛盾！命题得证。

10.2.2 范式

- E. 任何两属性关系模式必属于BCNF
  - 证明：设R(A1, A2)是两属性关系模式，则有4种非平凡函数依赖的情形：
    - (1)无任何非平凡的函数依赖：

      无冒犯BCNF 条件的函数依赖或R是全键，故R∈BCNF。
    - (2) A1→ A2，但A2 ↛ A1：

      Key=A1，唯一的决定子A1是超键，故R∈BCNF。
    - (3) A2→ A1，但A1 ↛ A2：

      Key=A2，唯一的决定子A2是超键，故R∈BCNF。
    - (4) A1 → A2，且A2 → A1：

      Key1=A1, Key2=A2，两个决定子均是超键，故R∈BCNF。

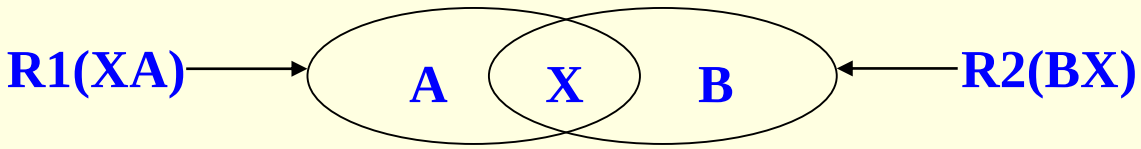
10.2.2 范式

- F.关系模式无损分解成BCNF的策略 ―启发式算法
  - 设关系模式R(XAB)，X, A, B均是R的属性(子集)，
    - (1) 针对冒犯BCNF 条件的某个非平凡函数依赖：

      $X \longrightarrow A$, (X非超键), 分解成两个模式:

      R1(XA), R2(BX)。

      一般地，R1∈BCNF，R2∉BCNF。
    - (2) 对不属于BCNF的模式R2(和R1)重复步骤(1)，直到全部属于 BCNF为止。



10.2.2 范式

- 例：M (title, year, length, color, star, star-gender, star-add, studio, studio-add, studio-class)
  - 根据通常的语义，有以下函数依赖：

    (1) star →star-gender, star-add

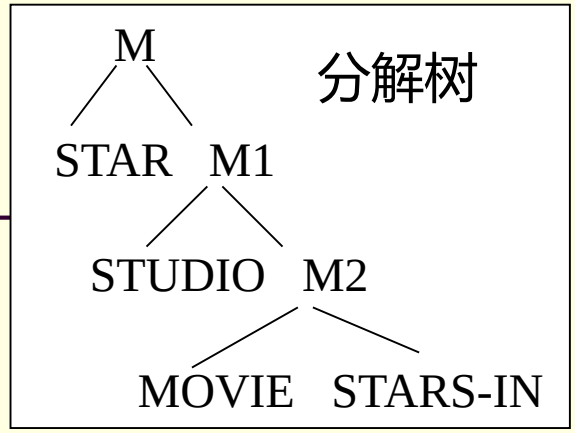
    (2) studio → studio-add, studio-class

    (3) title, year → length, color, studio

    故Key = {title, year, star}。

  因式(1), (2), (3)均冒犯BCNF 条件，故M ∉BCNF。

10.2.2 范式



- 模式分解
- 对式(1)，将M分解成：

  STAR (star, star-gender, star-add) ∈BCNF

  M1 (title, year, length, color, studio, studio-add, studio-class, star)

  因M1的Key未变，且式(2), (3)仍成立，故M1∉BCNF。
- 对式(2)，将M1分解成：

  STUDIO（studio, studio-add, studio-class）∈BCNF

  M2（title, year, length, color, star, studio）

  因M2的Key未变，且式(3)仍成立，故M2∉BCNF。
- 对式(3)，将M2分解成：

  MOVIE（title, year, length, color, studio）∈BCNF

  STARS-IN（title, year, star）∈BCNF。——全键关系模式。

  故，将M规范化成BCNF的一个无损分解为：

  ρ={STAR, STUDIO, MOVIE, STARS-IN}。

关系模式规范化的基本步骤

- 关系模式规范化的基本步骤

1NF

↓ 消除非主属性对键的部分函数依赖

2NF

↓ 消除非主属性对键的传递函数依赖

3NF

↓ 消除主属性对键部分和传递函数依赖

BCNF

目录 Contents

10.4 模式分解理论

- 分解的条件 / 准则
  - 无损分解——起码：决定能否分解；
  - 保持依赖分解——理想：决定分解的好坏。

- 结论
  - 总有将一个关系模式分解成3NF的无损、且保持依赖的分解。
  - 总有将一个关系模式分解成BCNF(甚至4NF)的无损的分解。