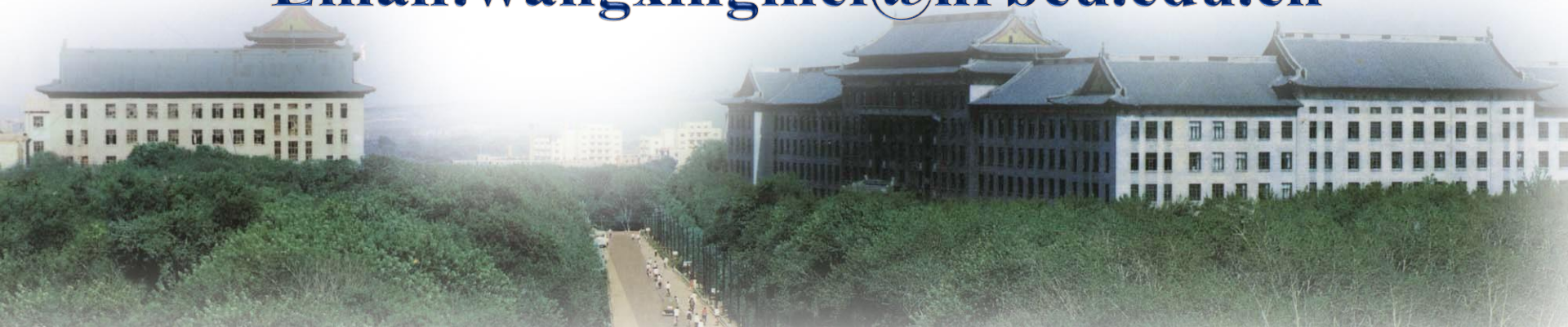
数据库原理

王兴梅
计算机科学与技术学院
Email:wangxingmei@hrbeu.edu.cn



第六章 关系数据理论

6.1 问题的提出

6.2 规范化

6.3 数据依赖的公理系统

*6.4 模式的分解

6.5 小结

6.1 问题的提出

关系数据库逻辑设计

- 针对具体问题，如何构造一个适合于它的数据模式
- 数据库逻辑设计的工具——关系数据库的规范化理论

问题的提出

一、概念回顾

二、关系模式的形式化定义

三、什么是数据依赖

四、关系模式的简化定义

五、数据依赖对关系模式影响

一、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尔积的一个子集。
- 关系模式：用来定义关系。
- 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。
- 关系数据库的模式：定义这组关系的关系模式的全体。

二、关系模式的形式化定义

关系模式由五部分组成，即它是一个五元组：

R(U, D, DOM, F)

R：　关系名

U：　组成该关系的属性名集合

D：　属性组U中属性所来自的域

DOM：属性向域的映象集合

F：　属性间数据的依赖关系集合

三、什么是数据依赖

1. 完整性约束的表现形式

- 限定属性取值范围：例如学生成绩必须在0-100之间
- 定义属性值间的相互关连（主要体现于值的相等与否），这就是数据依赖，它是数据库模式设计的关键

什么是数据依赖（续）

2. 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
- 是数据内在的性质
- 是语义的体现

什么是数据依赖（续）

3. 数据依赖的类型

- 函数依赖（Functional Dependency，简记为FD）
- 多值依赖（Multivalued Dependency，简记为MVD）
- 其他

四、关系模式的简化表示

- 关系模式R（U, D, DOM, F）

  简化为一个三元组：

  R（U, F）

- 当且仅当U上的一个关系r 满足F时，r称为关系模式 R（U, F）的一个关系

五、数据依赖对关系模式的影响

例：描述学校的数据库：

学生的学号（Sno）、所在系（Sdept）
系主任姓名（Mname）、课程名（Cname）
成绩（Grade）

单一的关系模式： Student <U、F>

$U = \{ Sno, Sdept, Mname, Cname, Grade \}$

数据依赖对关系模式的影响（续）

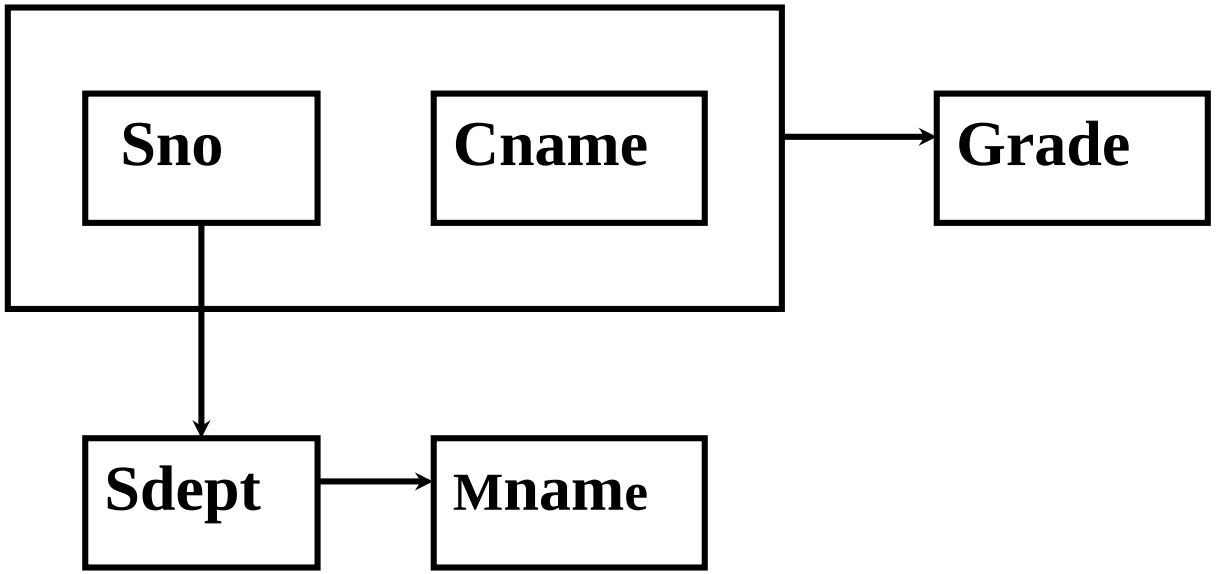
学校数据库的语义：

1. 一个系有若干学生， 一个学生只属于一个系；
2. 一个系只有一名主任；
3. 一个学生可以选修多门课程， 每门课程有若干学生选修；
4. 每个学生所学的每门课程都有一个成绩。

数据依赖对关系模式的影响（续）

属性组U上的一组函数依赖F：

F = { Sno → Sdept, Sdept → Mname,
(Sno, Cname) → Grade }



关系模式Student<U, F>中存在的问题

1. 数据冗余太大

   ❿ 浪费大量的存储空间

   例：每一个系主任的姓名重复出现

2. 更新异常（Update Anomalies）

   ➢ 数据冗余，更新数据时，维护数据完整性代价大。

   例：某系更换系主任后，系统必须修改与该系学生有关的每一个元组

关系模式Student<U, F>中存在的问题

3. 插入异常（Insertion Anomalies）

- 该插的数据插不进去

例，如果一个系刚成立，尚无学生，我们就无法把这个系及其系主任的信息存入数据库。

4. 删除异常（Deletion Anomalies）

- 不该删除的数据不得不删

例，如果某个系的学生全部毕业了， 我们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。

数据依赖对关系模式的影响（续）

结论：

- Student关系模式不是一个好的模式。
- “好”的模式：

不会发生插入异常、删除异常、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：通过分解关系模式来消除其中不合适的数据依赖。

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。

6.2.1 函数依赖

一、函数依赖

二、平凡函数依赖与非平凡函数依赖

三、完全函数依赖与部分函数依赖

四、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→Y。

X称为这个函数依赖的决定属性集(Determinant)。

说明：

1. 函数依赖不是指关系模式R的某个或某些关系实例满足的约束条件，而是指R的所有关系实例均要满足的约束条件。
2. 函数依赖是语义范畴的概念。只能根据数据的语义来确定函数依赖。

   例如“姓名→年龄”这个函数依赖只有在不允许有同名人的条件下成立
3. 数据库设计者可以对现实世界作强制的规定。例如规定不允许同名人出现，函数依赖“姓名→年龄”成立。所插入的元组必须满足规定的函数依赖，若发现有同名人存在， 则拒绝装入该元组。

函数依赖（续）

例: Student(Sno, Sname, Ssex, Sage, Sdept)

假设不允许重名，则有:

Sno → Ssex， Sno → Sage , Sno → Sdept，

Sno ←→ Sname, Sname → Ssex， Sname → Sage

Sname → Sdept

Ssex $\not\rightarrow$ Sage

若X→Y，并且Y→X，则记为X←→Y。

若Y不函数依赖于X， 则记为X $\not\rightarrow$ 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→Y，但Y $\not\subseteq$ X，则称X→Y是非平凡的函数依赖

若X→Y，但Y $\subseteq$ X，则称X→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) → Grade

平凡函数依赖：(Sno, Cno) → Sno

(Sno, Cno) → Cno

(Sno, Cno) → (Sno, Cno)

平凡函数依赖与非平凡函数依赖（续）

- 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。

三、完全函数依赖与部分函数依赖

定义6.2 在关系模式R(U)中，如果X→Y，并且对于X的任何一个真子集X'，都有

$X' \not\rightarrow Y$，则称Y完全函数依赖于X，记作$X \xrightarrow{F} Y$。

若X→Y，但Y不完全函数依赖于X，则称Y部分函数依赖于X，记作$X \xrightarrow{P} Y$。

完全函数依赖与部分函数依赖（续）

例: 在关系SC(Sno, Cno, Grade)中，

由于：Sno $\not\rightarrow$ Grade，Cno $\not\rightarrow$ Grade，

因此：(Sno, Cno) $\xrightarrow{F}$ Grade

在关系Student(Sno, Sname, Cno, Grade, Mname, Sdept)

由于：Sno $\rightarrow$ Sdept， Sno是(Sno, Cno) 的真子集

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

四、传递函数依赖

定义6.3 在关系模式R(U)中，如果X→Y，Y→Z，且Y ⊆X，Y↛X， Z ⊆Y，则称Z传递函数依赖于X。

注: 如果Y→X， 即X←→Y，则Z直接依赖于X。

例: 在关系Std(Sno, Sdept, Sloc)中，有：

Sno → Sdept，Sdept → Sloc

Sloc传递函数依赖于Sno

6.2.2 码

定义6.4 设K为关系模式R<U,F>中的属性或属性组合。

若$K \xrightarrow{F} U$，则K称为R的一个侯选码（Candidate Key）。

若关系模式R有多个候选码，则选定其中的一个做为主码（Primary key）。

- 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；不包含在任何候选码中的属性称为非主属性。

- 全码（ALL KEY）

举例

例: 在关系模式Student(Sno, Sname, Sdept, Cno, Grade)中，求关系模式S的候选码。

求解过程：（1）通过语义分析F

（2）提取满足完全函数确定关系的K

（3） 确定K是S的候选码

关系模式Student(Sno, Sname, Sdept, Cno, Grade)

根据语义分析得到F

F = { Sno → Sname, Sno → Sdept,
(Sno, Cno) → Grade }

确定满足完全函数确定关系的K

(Sno, Cno) 完全函数确定属性全集U

确定K为关系模式的候选码

(Sno, Cno) 为Student的候选码

外部码

定义6.5　关系模式 R 中属性或属性组X 并非R 的码，但 X 是另一个关系模式的码，则称 X 是R 的外部码（Foreign key）也称外码。

- 主码又和外部码一起提供了表示关系间联系的手段。

关系模式SC(Sno, Cno, Grade)中Sno不是码，但Sno是关系模式S(Sno, Sdept, Sage)的码，则Sno是关系模式SC的外码。

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∈nNF。

6.2.4 1NF

- 1NF的定义
  如果一个关系模式R的所有属性都是不可分的基本数据项，则R∈1NF。

- 第一范式是对关系模式的最起码的要求。不满足第一范式的数据库模式不能称为关系数据库。

- 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。

1NF

例: 关系模式　SLC(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 → Sdept

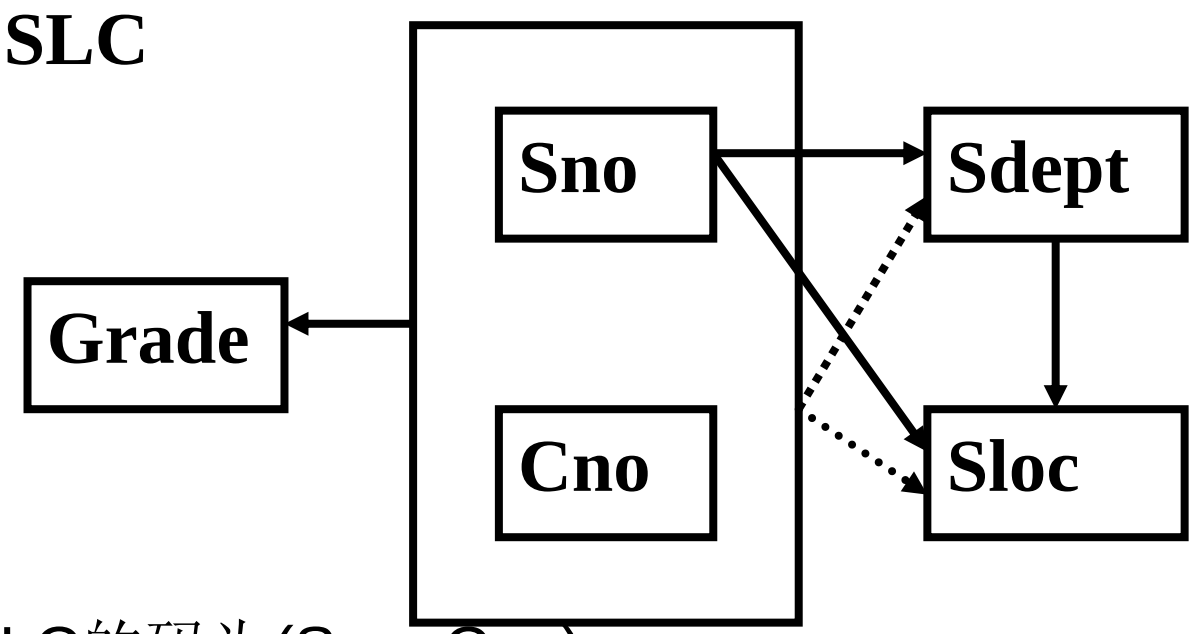
  (Sno, Cno) $\xrightarrow{P}$ Sdept

  Sno → Sloc

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

  Sdept → Sloc

1NF



- SLC的码为(Sno, Cno)
- SLC满足第一范式。
- 非主属性Sdept和Sloc部分函数依赖于码(Sno, Cno)

SLC不是一个好的关系模式

(1) 插入异常

假设Sno＝95102，Sdept＝IS，Sloc＝N的学生还未选课，因课程号是主属性，因此该学生的信息无法插入SLC。

(2) 删除异常

假定某个学生本来只选修了3号课程这一门课。现在因身体不适，他连3号课程也不选修了。因课程号是主属性，此操作将导致该学生信息的整个元组都要删除。

SLC不是一个好的关系模式

(3) 数据冗余度大

如果一个学生选修了10门课程，那么他的Sdept和Sloc值就要重复存储了10次。

(4) 修改复杂

例如学生转系，在修改此学生元组的Sdept值的同时，还可能需要修改住处（Sloc）。如果这个学生选修了K门课，则必须无遗漏地修改K个元组中全部Sdept、Sloc信息。

1NF

- 原因

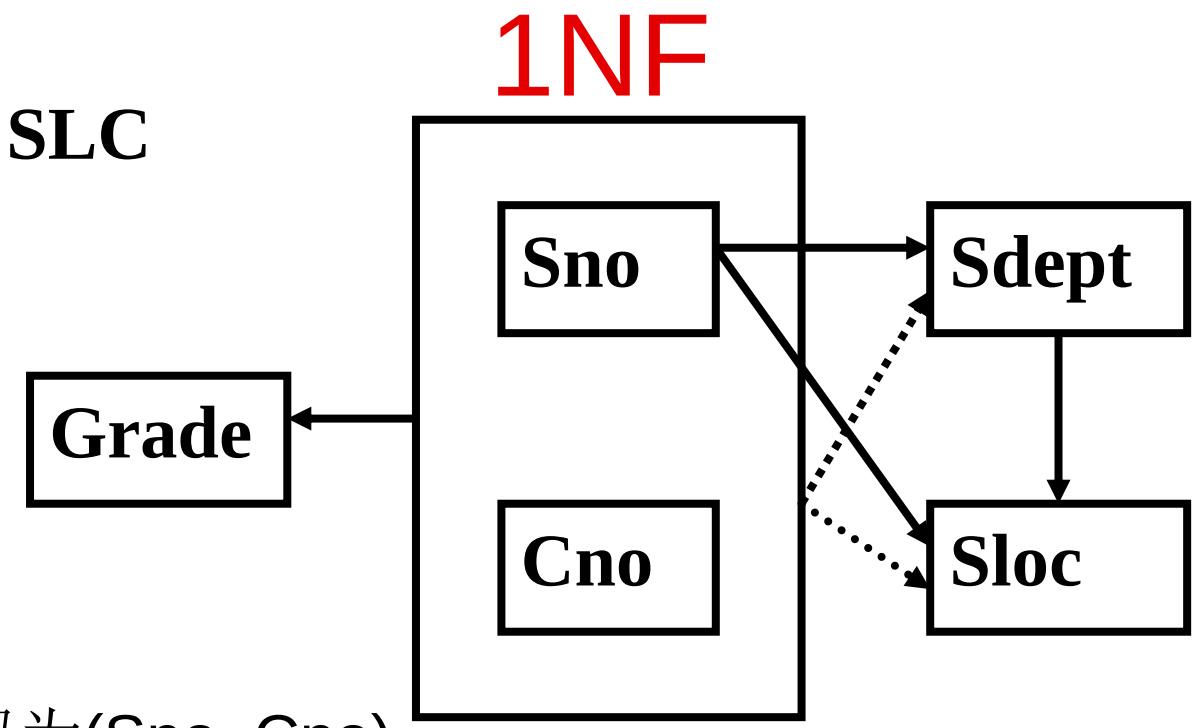
  Sdept、 Sloc部分函数依赖于码。

- 解决方法

SLC分解为两个关系模式，以消除这些部分函数依赖

SC（Sno， Cno， Grade）

SL（Sno， Sdept， Sloc）

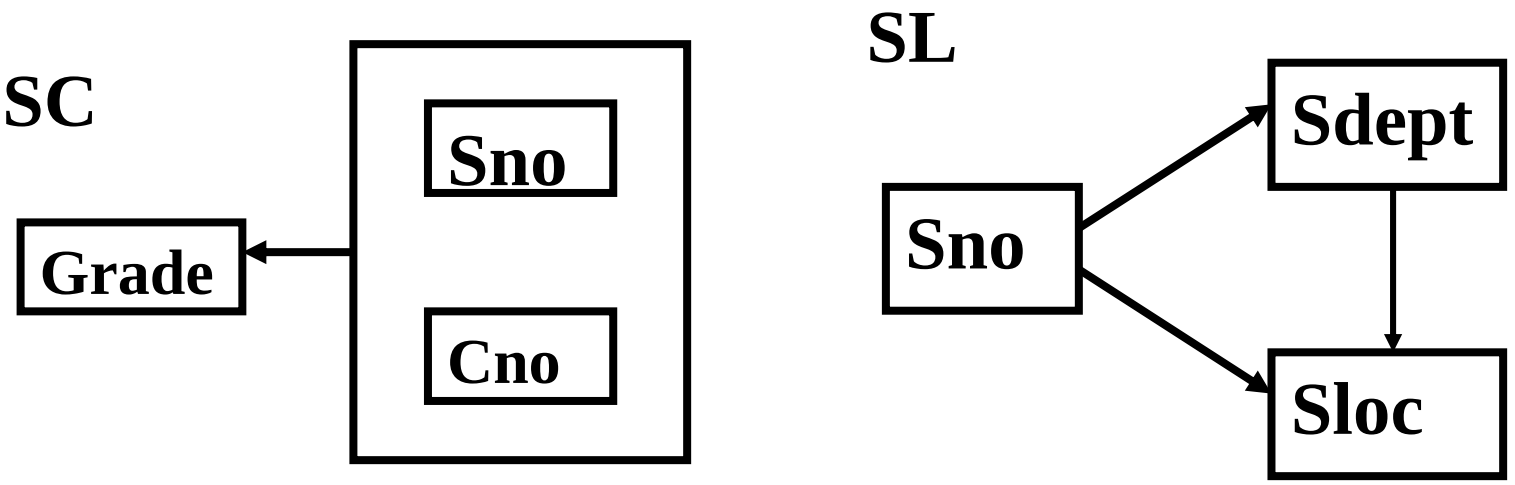


SLC的码为(Sno, Cno)

- SLC满足第一范式
- 非主属性Sdept和Sloc部分函数依赖于码(Sno, Cno)

1NF

函数依赖图：



2NF

- 2NF的定义

  定义5.6　若关系模式R∈1NF，并且每一个非主属性都完全函数依赖于R的码，则R∈2NF。

  例：SLC(Sno, Sdept, Sloc, Cno, Grade) ∈1NF

  SLC(Sno, Sdept, Sloc, Cno, Grade) ∉2NF

  SC（Sno， Cno， Grade） ∈ 2NF

  SL（Sno， Sdept， Sloc） ∈ 2NF

第二范式（续）

- 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。

6.2.5 3NF

例：2NF关系模式SL(Sno, Sdept, Sloc)中

- 函数依赖：

  Sno→Sdept

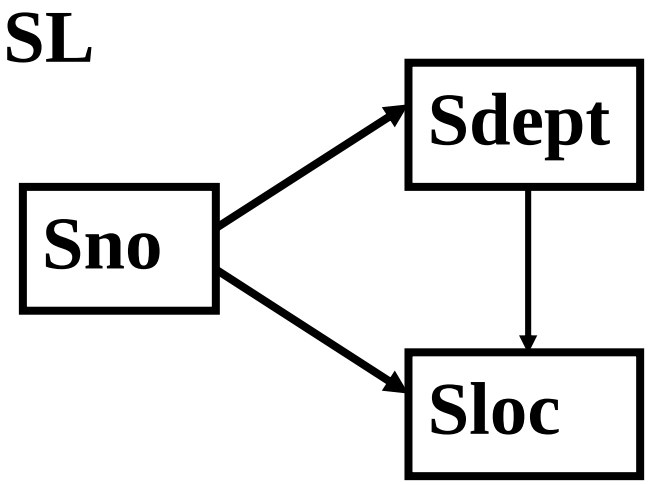
  Sdept→Sloc

  Sno→Sloc

Sloc传递函数依赖于Sno，即SL中存在非主属性对码的传递函数依赖。

3NF

函数依赖图：



3NF

- 解决方法

采用投影分解法，把SL分解为两个关系模式，以消除传递函数依赖：

SD（Sno， Sdept）

DL（Sdept， Sloc）

SD的码为Sno， DL的码为Sdept。

3NF

SD的码为Sno， DL的码为Sdept。

SD **DL**

3NF

- 3NF的定义

  定义5.8 关系模式$R<U，F>$ 中若不存在这样的码X、属性组Y及非主属性Z（$Z \nsubseteq Y$），使得$X \rightarrow Y$，$Y \nrightarrow X$，$Y \rightarrow Z$，成立，则称$R<U，F> \in 3NF$。

即 若$R<U，F> \in 2NF$，不存在非主属性对候选码的传递函数依赖，则$R<U，F> \in 3NF$。

例， SL(Sno, Sdept, Sloc) ∈ 2NF

SL(Sno, Sdept, Sloc) ∉ 3NF

SD（Sno， Sdept） ∈ 3NF

DL（Sdept， Sloc）∈ 3NF

3NF

- 若R∈3NF，则R的每一个非主属性既不部分函数依赖于候选码也不传递函数依赖于候选码。
- 如果R∈3NF，则R也是2NF。
- 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。

6.2.6 BC范式（BCNF）

- 定义6.9 设关系模式R<U，F>∈1NF，如果对于R的每个函数依赖X→Y，若Y不属于X，则X必含有候选码，那么R∈BCNF。

若R∈BCNF

- 每一个决定属性集（因素）都包含（候选）码
- R中的所有属性（主，非主属性）都完全函数依赖于码
- R∈3NF（证明）
- 若R∈3NF 则 R不一定∈BCNF

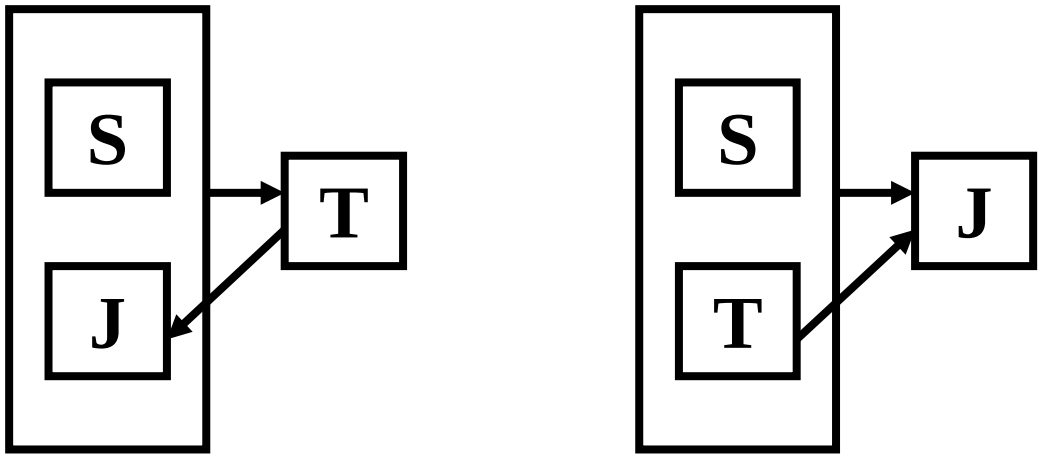
BCNF

例：在关系模式STJ（S，T，J）中，S表示学生，T表示教师，J表示课程。

- 每一教师只教一门课。每门课由若干教师教，某一学生选定某门课，就确定了一个固定的教师。某个学生选修某个教师的课就确定了所选课的名称：

  (S，J)→T，(S，T)→J，T→J

6.2.6 BCNF



STJ

BCNF

STJ∈3NF

- (S，J)和(S，T)都可以作为候选码
- S、T、J都是主属性

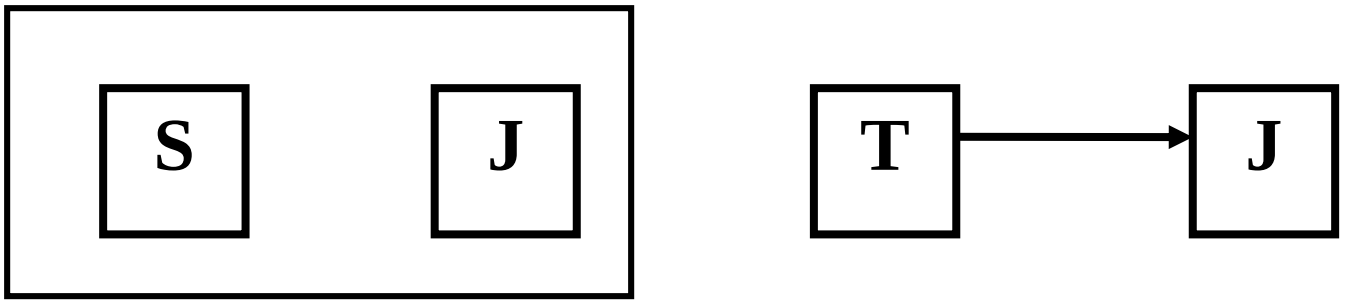
STJ∉BCNF

- T→J，T是决定属性集，T不是候选码

BCNF

解决方法：将STJ分解为二个关系模式：

SJ(S，J) ∈ BCNF， TJ(T，J)∈ BCNF



ST **TJ**

没有任何属性对码的部分函数依赖和传递函数依赖

3NF与BCNF的关系

- 如果关系模式R∈BCNF，
  必定有R∈3NF
- 如果R∈3NF，且R只有一个候选码，
  则R必属于BCNF。

BCNF的关系模式所具有的性质

1. 所有非主属性都完全函数依赖于每个候选码

2. 所有主属性都完全函数依赖于每个不包含它的候选码

3. 没有任何属性完全函数依赖于非码的任何一组属性

6.2 规范化

6.2.1 第一范式（1NF）

6.2.2 第二范式（2NF）

6.2.3 第三范式（3NF）

6.2.4 BC范式（BCNF）

6.2.5 多值依赖与第四范式（4NF）

6.2.6 规范化

6.2.6 规范化

- 关系数据库的规范化理论是数据库逻辑设计的工具。
- 一个关系只要其分量都是不可分的数据项，它就是规范化的关系，但这只是最基本的规范化。
- 规范化程度可以有多个不同的级别

规范化（续）

- 规范化程度过低的关系不一定能够很好地描述现实世界，可能会存在插入异常、删除异常、修改复杂、数据冗余等问题
- 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集合，这种过程就叫**关系模式的规范化**

规范化（续）

- 关系模式规范化的基本步骤

消除决定属性集非码的非平凡函数依赖

1NF
↓ 消除非主属性对码的部分函数依赖
2NF
↓ 消除非主属性对码的传递函数依赖
3NF
↓ 消除主属性对码的部分和传递函数依赖
BCNF
↓ 消除非平凡且非函数依赖的多值依赖
4NF

规范化的基本思想

- 消除不合适的数据依赖
- 各关系模式达到某种程度的“分离”
- 采用“一事一地”的模式设计原则

  让一个关系描述一个概念、一个实体或者实体间的一种联系。若多于一个概念就把它“分离”出去
- 所谓规范化实质上是概念的单一化

规范化（续）

- 不能说规范化程度越高的关系模式就越好
- 在设计数据库模式结构时，必须对现实世界的实际情况和用户应用需求作进一步分析，确定一个合适的、能够反映现实世界的模式
- 上面的规范化步骤可以在其中任何一步终止

设W（C,P,S,G,T,R）其中C为课程，P为教师，S为学生，G为成绩，T为时间，R为教室。存在有如下函数依赖集

{ (S,C)->G,(T,R)->C,(T,P)->R,(T,S)->R }

关系模式W的候选码_T，S，P_，W的规范化程度最高达到_1NF_。若将W分解为3个关系模式W1(C,P)，W2(S,C,G)和W3(S,T,R,C)则W1规范化程度最高达到_BCNF_，W2规范化程度_BCNF_，W3规范化程度_2NF_。

- 设关系模式R(A,B,C,D,E,P)，其中F={A->B,C->P,E->A,CE->D}求R的候选码。
- CE
- 设R(C,T,S,N,G)，其中F={C->T,CS->G,S->N}求R的候选码。
- CS

第六章 关系数据理论

6.1 数据依赖

6.2 规范化

6.3 数据依赖的公理系统

6.4 模式的分解

6.3 数据依赖的公理系统

- 逻辑蕴含

定义5.11 对于满足一组函数依赖 F 的关系模式$R<U，F>$，其任何一个关系r，若函数依赖$X \rightarrow Y$都成立, 则称

F逻辑蕴含$X \rightarrow Y$

Armstrong公理系统

- 一套推理规则，是模式分解算法的理论基础
- 用途
  - 求给定关系模式的码
  - 从一组函数依赖求得蕴含的函数依赖

1. Armstrong公理系统

关系模式$R<U, F>$来说有以下的推理规则：

- Al.**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。

注意：由自反律所得到的函数依赖均是平凡的函数依赖，自反律的使用并不依赖于F

定理 6.I Armstrong推理规则是正确的

（I）自反律:若$Y \subseteq X \subseteq U$，则$X \rightarrow Y$为$F$所蕴含

证: 设$Y \subseteq X \subseteq U$

对$R<U, F>$的任一关系r中的任意两个元组t，s：

若$t[X]=s[X]$，由于$Y \subseteq X$，有$t[Y]=s[Y]$，

所以$X \rightarrow Y$成立.

自反律得证

注：$t[X]$表示元组t在属性（组）X上的分量，等价于$t.[X]$

定理6.1

(2)增广律： 若$X\rightarrow Y$为F所蕴含，且$Z \subseteq U$，则$XZ\rightarrow YZ$ 为F所蕴含。

证：设$X\rightarrow Y$为F所蕴含，且$Z \subseteq U$。

设$R<U，F>$ 的任一关系r中任意的两个元组t，s；

若$t[XZ]=s[XZ]$，则有$t[X]=s[X]$和$t[Z]=s[Z]$；

由$X\rightarrow Y$，于是有$t[Y]=s[Y]$，所以$t[YZ]=s[YZ]$，所以$XZ\rightarrow YZ$为F所蕴含.

增广律得证。

定理6.I

(3) 传递律：若$X \rightarrow Y$及$Y \rightarrow Z$为F所蕴含，则$X \rightarrow Z$为 F所蕴含。

证：设$X \rightarrow Y$及$Y \rightarrow Z$为F所蕴含。

对$R<U, F>$ 的任一关系 r中的任意两个元组 t，s。

若$t[X]=s[X]$，由于$X \rightarrow Y$，有 $t[Y]=s[Y]$；

再由$Y \rightarrow Z$，有$t[Z]=s[Z]$，所以$X \rightarrow Z$为F所蕴含.

传递律得证。

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.I $X \rightarrow A_1 A_2 \ldots A_k$成立的充分必要条件是$X \rightarrow A_i$成立（$i$=I，2，…，$k$）。

3. 函数依赖闭包

定义6.l2　在关系模式$R<U，F>$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，$X \subseteq U$，　$X_F^+ =\{ A|X \rightarrow A$能由$F$ 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。

F的闭包

F={X→Y,Y→Z}, X→A1A2...An，F^+计算是NP完全问题

F^+={

X→φ, Y→φ, Z→φ, XY→φ, XZ→φ, YZ→φ, XYZ→ φ,

X→X, Y→ Y, Z→Z, XY→ X, XZ→X, YZ→Y, XYZ→ X,

X→Y, Y→Z , XY →Y, XZ→Y, YZ →Z, XYZ →Y,

X→Z, Y→YZ, XY→Z, XZ→Z, YZ→YZ, XYZ →Z,

X→XY, XY→XY, XZ→XY, XYZ →XY,

X→XZ, XY→YZ, XZ→XZ, XYZ→YZ

X→YZ, XY →XZ, XZ→XY, XYZ→XZ,

X→ZYZ, XY →XYZ, XZ→XYZ, XYZ→XYZ }

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，就转化为求出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.l 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$B=\{A \mid (\exists V)(\exists W)(V \rightarrow W \in F \wedge V \subseteq X^{(i)} \wedge A \in W)\}$；

（3）$X^{(i+1)}=B \cup X^{(i)}$

算法6.l

（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。

对于算法6.l，令$a_i=|X^{(i)}|$，$\{a_i\}$形成一个步长大于1的严格递增的序列，序列的上界是$|U|$，因此该算法最多$|U|-|X|$次循环就会终止。

函数依赖闭包

[例] 已知关系模式$R<U，F>$，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)计算$X^{(1)}$：逐一的扫描F集合中各个函数依赖，找左部为A，B或AB的函数依赖。得到两个：

$AB\rightarrow C，B\rightarrow D$。

于是$X^{(1)}=AB\cup CD=ABCD$。

函数依赖闭包

(2)因为$X^{(0)} \neq X^{(1)}$，所以再找出左部为*ABCD*子集的那些函数依赖，又得到$AB \rightarrow C$，$B \rightarrow D$，$C \rightarrow E$，$AC \rightarrow B$，于是$X^{(2)}=X^{(1)} \cup BCDE=ABCDE$。

(3)因为$X^{(2)}$=U，算法终止

所以$(AB)_F^+ =ABCDE$。

设R（U，F），其中U={A,B,C,D,E,G},
F={AB->C,C->A,BC->D,ACD->B,
D->EG,BE->C, CG->BD,CE->AG}
求$(BD)^+$
$(BD)^+$ =U

5. 函数依赖集等价

定义6.14 如果$G^+=F^+$，就说函数依赖集F覆盖G（F是G的覆盖，或G是F的覆盖），或F与G等价。

函数依赖集等价的充要条件

引理6.3 $F^+ = G^+$ 的充分必要条件是

$$F \subseteq G^+ \text{，和} G \subseteq F^+$$

证：必要性显然，只证充分性。

（1）若$F \subseteq G^+$，则$X_F{}^+ \subseteq X_{G+}{}^+$。

（2）任取$X \rightarrow Y \in F^+$ 则有 $Y \subseteq X_F{}^+ \subseteq X_{G+}{}^+$。

所以$X \rightarrow Y \in (G^+)^+ = G^+$。即$F^+ \subseteq G^+$。

（3）同理可证$G^+ \subseteq F^+$，所以$F^+ = G^+$。

函数依赖集等价

- 要判定$F \subseteq G^+$，只须逐一对F中的函数依赖$X \rightarrow Y$，考察Y是否属于X_{G+}就行了。因此引理6.3给出了判断两个函数依赖集等价的可行算法。

6. 最小函数依赖集

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。

化简函数依赖

(1) F中任一函数依赖的右部仅含有一个属性。

除去冗余函数依赖

(2) F中不存在这样的函数依赖X→A，使得F与F-{X→A}等价。

除去冗余的决定因子

(3) F中不存在这样的函数依赖X→A，　X有真子集Z使得F-{X→A}∪{Z→A}与F等价。

最小依赖集

[例] 对于6.l节中的关系模式*S*<*U*，*F*>，其中：

U={ SNO，SDEPT，MN，CNAME，G }，

F={ SNO→SDEPT ， SDEPT→MN ， （ SNO ，CNAME）→G }

设F’ ={SNO→SDEPT，SNO→MN，SDEPT→MN，(SNO，CNAME)→G， (SNO，SDEPT)→SDEPT}

*F*是最小覆盖（最小依赖集），而*F* ’不是。

因为：*F* ’-{SNO→MN}与*F* ’等价

F ’-{(SNO，SDEPT)→SDEPT}也与*F* ’等价

F ’-{(SNO，SDEPT)→SDEPT}

∪{SNO→SDEPT}也与*F* ’等价

7. 极小化过程

定理6.3 每一个函数依赖集F均等价于一个极小函数依赖集F_m。此F_m称为F的最小依赖集

证:构造性证明，依据定义分三步对F进行“极小化处理”，找出F的一个最小依赖集。

(1)逐一检查F中各函数依赖FD_i：$X \rightarrow Y$，

若$Y=A_1A_2 \ldots A_k$，$k > 2$，

则用 $\{ X \rightarrow A_j | j=1，2，\ldots，k\}$ 来取代$X \rightarrow Y$。

引理6.1保证了F变换前后的等价性。

极小化过程

(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。

由于F与$G=F-\{X \rightarrow A\}$等价的充要条件是$A \in X_G^+$

因此F变换前后是等价的。

极小化过程

(3)逐一取出F中各函数依赖FD_i：$X \rightarrow A$，

设$X=B_1B_2\ldots B_m$，

逐一考查B_i（i=l，2，…，m），

若$A \in (X-B_i)_F^+$，

则以$X-B_i$取代X。

由于F与$F-\{X \rightarrow A\} \cup \{Z \rightarrow A\}$等价的充要条件是$A \in Z_F^+$，其中$Z=X-B_i$

因此F变换前后是等价的。

极小化过程

由定义，最后剩下的*F*就一定是极小依赖集。

因为对*F*的每一次“改造”都保证了改造前后的两个函数依赖集等价，因此剩下的*F*与原来的*F*等价。 证毕

- 定理6.3的证明过程 也是求*F*极小依赖集的过程

极小化过程

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

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不一定是唯一的它与对各函数依赖FD_i及$X \rightarrow A$中X各属性的处置顺序有关

极小化过程

- 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
  - 若改造后的*F*与原来的*F*相同，说明*F*本身就是一个最小依赖集

极小化过程

- 在$R<U, F>$中可以用与F等价的依赖集G来取代F
  - 原因：两个关系模式$R_1<U, F>$，$R_2<U, G>$，如果F与G等价，那么R_1的关系一定是R_2的关系。反过来，R_2的关系也一定是R_1的关系。

设有R(A,B,C,D) ,F={A->C,C->A,B->AC,D->AC}

1.计算（AD）$^+$

2.求F的最小等价依赖集Fm

（AD）$^+$=ADC

Fm={A->C,C->A,B->C,D->C}

解法1：

1.最小函数依赖集第一步，右侧变单属性：

F={ A->C, C->A, B->A, B->C, D->A ,D->C}

2.判断冗余函数依赖：

- 若去掉A->C,F’={C->A, B->A, B->C, D->A, D->C}

F’与F不等价，所以A->C是必要的。

- 若去掉C->A,同理可得到C->A是必要的。
- 若去掉B->A,F’={A->C, C->A, B->C, D->A, D->C}

由B->C,C->A得到B->A，F’与F等价，B->A冗余，直接去掉。

- 若去掉B->C,　同理可得到B->C是必要的。
- 若去掉D->A,　F'={A->C, C->A, B->C, D->C}

由D->C,C->A…得到D->A冗余。

- 若去掉D->C,　同理可得到D->C是必要的。

3.综上所述，Fm_1={A->C,C->A,B->C,D->C}

解法2：

1.最小函数依赖集第一步，右侧变单属性：

F={A->C, C->A, B->A, B->C, D->A ,D->C}

2.判断冗余函数依赖：

- 若去掉D->C,F’={A->C, C->A, B->A, B->C, D->A,}

F’与F等价，D->C冗余，去掉。

- 若去掉D->A,…可得到D->A是必要的。
- 若去掉B->C,…F’={A->C, C->A, B->A, D->A,}

F’与F等价，B->C冗余，去掉。

- 若去掉B->A,…可得到B->A是必要的
- 若去掉C->A,…可得到C->A是必要的
- 若去掉A->C,…可得到A->C是必要的

3.综上所述，Fm_2={A->C,C->A,B->A,D->A}

第六章 关系数据理论

6.1 数据依赖

6.2 规范化

6.3 数据依赖的公理系统

6.4 模式的分解

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并不是唯一的
- 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

三种模式分解的等价定义

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

模式的分解（续）

定义6.16 关系模式R<U,F>的一个分解：

$\rho=\{ R_1<U_1,F_1>，R_2<U_2,F_2>，\ldots，R_n<U_n,F_n>\}$

$U=U_1 \cup U_2 \cup \ldots \cup U_n$，且不存在 $U_i \subseteq U_j$，F_i 为 F在 U_i 上的投影

定义6.17 函数依赖集合$\{X \rightarrow Y \mid X \rightarrow Y \in F^+ \wedge XY \subseteq U_i\}$

的一个覆盖 F_i 叫作 F 在属性 U_i 上的投影

模式的分解（续）

例：SL（Sno， Sdept， Sloc）

F={ Sno→Sdept,Sdept→Sloc,Sno→Sloc}

SL∈2NF

存在插入异常、删除异常、冗余度大和修改复杂等问题

分解方法可以有多种

模式的分解（续）

SL

Sno	Sdept	Sloc
95001	CS	A
95002	IS	B
95003	MA	C
95004	IS	B
95005	PH	B

模式的分解（续）

1. SL分解为下面三个关系模式：

SN(Sno)

SD(Sdept)

SO(Sloc)

分解后的关系为：

SN

Sno
95001
95002
95003
95004
95005

SD

Sdept
CS
IS
MA
PH

SO

Sloc
A
B
C

模式的分解（续）

分解后的数据库丢失了许多信息

例如无法查询95001学生所在系或所在宿舍。

如果分解后的关系可以通过自然连接恢复为原来的关系，那么这种分解就没有丢失信息

模式的分解（续）

2. SL分解为下面二个关系模式：

NL(Sno, Sloc)

DL(Sdept, Sloc)

分解后的关系为：

NL

Sno	Sloc
95001	A
95002	B
95003	C
95004	B
95005	B

DL

Sdept	Sloc
CS	A
IS	B
MA	C
PH	B

模式的分解（续）

NL ⋈ DL

Sno	Sloc	Sdept
95001	A	CS
95002	B	IS
95002	B	PH
95003	C	MA
95004	B	IS
95004	B	PH
95005	B	IS
95005	B	PH

模式的分解（续）

NL⋈DL比原来的SL关系多了3个元组

无法知道95002、95004、95005

究竟是哪个系的学生

元组增加了，信息丢失了

第三种分解方法

3. 将SL分解为下面二个关系模式：

ND(Sno, Sdept)

NL(Sno, Sloc)

分解后的关系为：

模式的分解（续）

ND

Sno	Sdept
95001	CS
95002	IS
95003	MA
95004	IS
95005	PH

NL

Sno	Sloc
95001	A
95002	B
95003	C
95004	B
95005	B

模式的分解（续）

ND⋈ NL

Sno	Sdept	Sloc
95001	CS	A
95002	IS	B
95003	MA	C
95004	CS	A
95005	PH	B

与SL关系一样，因此没有丢失信息

具有无损连接性的模式分解

- 关系模式R<U,F>的一个分解 ρ={ R_1<U_1,F_1>，R_2<U_2,F_2>， …，R_n<U_n,F_n>}

若R与R1、R2、…、Rn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

模式的分解（续）

第三种分解方法具有无损连接性

问题:

这种分解方法没有保持原关系中的函数依赖

SL中的函数依赖Sdept→Sloc

没有投影到关系模式ND、NL上

保持函数依赖的模式分解

设关系模式R<U,F>被分解为若干个关系模式

$R_1<U_1,F_1>$，$R_2<U_2,F_2>$，…，$R_n<U_n,F_n>$

（其中$U=U_1 \cup U_2 \cup \ldots \cup U_n$，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）。

第四种分解方法

将SL分解为下面二个关系模式：

ND(Sno, Sdept)

DL(Sdept, Sloc)

这种分解方法就保持了函数依赖。

模式的分解（续）

- 如果一个分解具有无损连接性，则它能够保证不丢失信息。
- 如果一个分解保持了函数依赖，则它可以减轻或解决各种异常情况。
- 分解具有无损连接性和分解保持函数依赖是两个互相独立的标准。具有无损连接性的分解不一定能够保持函数依赖。同样，保持函数依赖的分解也不一定具有无损连接性。

模式的分解（续）

第一种分解方法既不具有无损连接性，也未保持函数依赖，它不是原关系模式的一个等价分解

第二种分解方法保持了函数依赖，但不具有无损连接性

第三种分解方法具有无损连接性，但未持函数依赖

第四种分解方法既具有无损连接性，又保持了函数依赖

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6.4 转换为3NF既有无损连接性又保持函数依赖的分解
- 算法6.5 转换为BCNF的无损连接分解（分解法）
- 算法6.6 达到4NF的具有无损连接性的分解

算法6.2 判别一个分解的无损连接性

1. 建立一个n列k行的表。填入a_i，或b_{ij}。
2. 对每个函数依赖做下列操作：找到Xi所对应的列中具有相同符号那些行，若其中有a_i，则全部改成a_i；否则全部改为行号最小的b_{ij}。若某个b_{ij}被更改，那么该表中其它相同b_{ij}均做相同的更改。
3. 比较扫描后有无变化，无变化则终止。若表中有全a行，则分解具有无损连接性。

例： 已知R<U，F>，U={A,B,C,D,E},F={A->C,B->C,C->D,DE->C,CE->A}

判断R的一个分解

$R_1(A,D),R_2(B,C),R_3(B,E),R_4(C,D,E),R_5(A,E)$是否是无损连接分解？

(1)建原始表

	A	B	C	D	E
AD	a_1	b_{12}	b_{13}	a_4	b_{15}
BC	b_{21}	a_2	a_3	b_{24}	b_{25}
BE	b_{31}	a_2	b_{33}	b_{34}	a_5
CDE	b_{41}	b_{42}	a_3	a_4	a_5
AE	a_1	b_{52}	b_{53}	b_{54}	a_5

(2)根据依赖关系修改原始表

对于A->C，A列中第1行和第5行相等都是a_1，那么C列中对应的两行也应该相等（b_{13}和b_{53}应相等），将b_{53}改为标号小的b_{13}。

对于B->C，B列中第2行和第3行相等都是a_2，那么C列中对应的两行也应该相等（ a_3和b_{33}应相等），a的优先级比b高，将b_{33}改为a_3。

	A	B	C	D	E
AD	a_1	b_{12}	b_{13}	a_4	b_{15}
BC	b_{21}	a_2	a_3	b_{24}	b_{25}
BE	b_{31}	a_2	b_{33}	b_{34}	a_5
CDE	b_{41}	b_{42}	a_3	a_4	a_5
AE	a_1	b_{52}	b_{53}	b_{54}	a_5

	A	B	C	D	E
AD	a_1	b_{12}	b_{13}	a_4	b_{15}
BC	b_{21}	a_2	a_3	b_{24}	b_{25}
BE	b_{31}	a_2	a_3	b_{34}	a_5
CDE	b_{41}	b_{42}	a_3	a_4	a_5
AE	a_1	b_{52}	b_{13}	b_{54}	a_5

对于C->D，C列中第1行和第5行相等都是b_{13}，那么D列中对应的两行也应该相等（a_4和b_{54}应相等），a的优先级比b高，将b_{54}改为a_4。

另外，C列中第2行、第3行和第4行相等都是a_3，那么D列中对应的行也应该相等（ b_{24}、 b_{34}和a_4应相等），a的优先级比b高，将b_{24}和b_{34}改为a_4。

	A	B	C	D	E
AD	a_1	b_{12}	b_{13}	a_4	b_{15}
BC	b_{21}	a_2	a_3	a_4	b_{25}
BE	b_{31}	a_2	a_3	a_4	a_5
CDE	b_{41}	b_{42}	a_3	a_4	a_5
AE	a_1	b_{52}	$b_{13}(a_3)$	a_4	a_5

对于DE->C，DE列中公共的相等行为第3行、第4行和第5行，那么C列中对应的行也应该相等（a_3和b_{13}应相等），a的优先级比b高，将b_{13}改为a_3。

对于CE->A，CE列中公共的相等行为第3行、第4行和第5行相等都是a_3，那么A列中对应的行也应该相等（ b_{31}、b_{41}和a_1应相等），a的优先级比b高，将b_{31}和b_{41}改为a_1。

	A	B	C	D	E
AD	a_1	b_{12}	b_{13}	a_4	b_{15}
BC	b_{21}	a_2	a_3	a_4	b_{25}
BE	a_1	a_2	a_3	a_4	a_5
CDE	a_1	b_{42}	a_3	a_4	a_5
AE	a_1	b_{52}	a_3	a_4	a_5

(3)直到表格中有一行为a则为无损连接

扫描一遍表格没有变化则停止，有变化，但有一行不全是a则继续扫描。

最后，从表格中看BE的行都为a，R的一个分解R1(A,D),R2(B,C),R3(B,E),R4(C,D,E),R5(A,E)是无损连接分解。

- 例：R(A,B,C)　F={A->B,C->B}
- ρ 1=(AB, BC)
- ρ 1分解不具有无损连接性
- ρ 2=(AC, BC)
- ρ 2分解具有无损连接性
- ρ 3=(AC, AB)
- ρ 3分解具有无损连接性

- 例： R<U,F>,U={A,B,C,D,E},F={AB->C,C->D,D->E}
- ρ={R1(ABC),R2(CD),R3(DE)}
- 具有无损连接性

判别一个分解的函数依赖保持性

输入：关系模式R上的函数依赖集F，
　　R的一个分解：ρ={R1，R2・・・Rk}
输出：确定ρ是否具有依赖保持性
方法：
1.计算F到每一个Ri上的投影（i=1…k）
2.FOR每一个X-〉Y∈F DO
　Z1=X；Z0=Φ；
　DO WHILE Z1≠Z0
　　Z0=Z1；
　　FOR i=1 TO k DO
　　Z1=Z1∪ ((Z1 ∩Ri)$^{+}$ ∩Ri)
　　ENDFOR
　ENDDO
　IF　Y-Z1= Φ　RETURN(true)
 ENDFOR
3.RETURN(false)

- 例：R<U,F>,U={A,B,C,D},F={A->B,B->C,C->D,D->A}判定分解ρ={AB, BC, CD}是否具有函数依赖保持性

回顾知识点

定义6.14 如果$G^+=F^+$，就说函数依赖集F覆盖G（F是G的覆盖，或G是F的覆盖），或F与G等价。

引理6.3 $F^+ = G^+$ 的充分必要条件是

$$F \subseteq G^+，和G \subseteq F^+$$

总结 要判定$F \subseteq G^+$，只须逐一对F中的函数依赖$X \rightarrow Y$，考察 Y 是否属于X_{G+}^{+} 就行了。

解：

F1(AB)= ={A->B,B->A} 在F中找到所有仅包含A,B的函数依赖关系

F2(BC)= ={B->C,C->B} 在F中找到所有仅包含B,C的函数依赖关系

F3(CD)= ={C->D,D->C} 在F中找到所有仅包含C,D的函数依赖关系

G=F1UF2UF3={A->B, B->A,B->C,C->B,C->D,D->C}

易知$G \subseteq F^{+}$

判别$F \subseteq G^+$，即判定D->A $\in G^+$

$D_{G+}{}^+$ 为DCBA

所以$G^+=F^+$，即分解后G与F等价，所以保持了FD。

Student(Sno,Sdept,Sboss)， F={Sno->Sdept，Sdept->Sboss}

判定下列分解是否具有无损连接性和函数依赖保持性

若分解为三张表SN(Sno) SD(Sdept) SO(Sboss)

若分解为二张表SD(Sno, Sdept) SS(Sno, Sboss)

若分解为二张表SD(Sno, Sdept) DS(Sdept, Sboss)

算法6.3 （合成法）转换为3NF的保持函数依赖的分解。

方法：

1.如果F’中有一个函数依赖X->A，且XA=R，则输出ρ{R}，转4。

2.如果R中某些属性与F’中所有函数依赖的左部和右部都无关，则将它们构成一个关系模式，从R中将它们分离出去。

3.对于F’中每一个Xi-> Ai都构成一个关系模式Ri=XiAi。但是当X-> A1，…An均属于F时，则使用模式XA1…An来代替XAi。

4.停止分解，输入ρ。

- 例：
  S(Sno,Sname,Sage,Ssex,Sdept,Deptname,Cno,Grade)

  F={Sno->Sname,Sno->Sage,Sno->Ssex,Sno->Sdept,Sdept->Deptname,(Sno,Cno)->Grade}

ρ={(Sno, Sname, Sage, Ssex, Sdept), (Sdept, Deptname), (Sno, Cno, Grade)}

转换为3NF的保持函数依赖的分解：

S(Sno,Sname,Sage,Ssex,Sdept,Departname,Cno,Grade)

步骤：

1.计算F的最小函数依赖集

a.所有函数依赖变成右边都是单个属性的函数依赖。

这里F中右边都是单个属性，所以不用分解。

b.去掉F中多余的函数依赖

F={Sno->Sname, Sno->Sage, Sno->Ssex, Sno->Sdept, Sdept->Departname,(Sno,Cno)->Grade}

设Sno->Sname为冗余的函数依赖，去掉后为：

F1={Sno->Sage, Sno->Ssex, Sno->Sdept, Sdept->Departname, (Sno, Cno)->Grade}

计算$(Sno)_F^+$:设X(0)=Sno扫描剩余的左部有Sno的。Sno->Sage，Sno->Ssex，Sno->Sdept推不出Sname，故不冗余，不能将Sno->Sname去掉。

设Sno->Sage为冗余的函数依赖，去掉后为：

F2={Sno->Sname, Sno->Ssex, Sno->Sdept, Sdept->Departname,(Sno,Cno)->Grade}

同理，推不出Sage，故也不冗余，不能将Sno->Sage去掉。

……

c.所有函数依赖变成左边没有多余的属性的函数依赖。

这里F中左边不存在多余的属性。

2.将未出现在Fm中的属性取出来变成一个关系模式

S中所有属性都在F中出现。

3.Fm中每一个函数依赖，左侧和右侧属性组合成一个关系模式

由于S中所有属性都在F中出现，所以最小函数依赖集不变。

F={Sno->Sname,Sno->Sage,Sno->Ssex,Sno->Sdeapt,Sdept->Departname,(Sno,Cno)->Grade}

4.将F中由相同左侧属性的多个函数依赖进行合并作为一个分解

R1={Sno,Sname,Sage,Ssex,Sdept}

R2={Sdept,Department}

R3={Sno,Cno,Grade}

故分解结果为：

ρ={R1,R2,R3}

ρ={(Sno,Sname,Sage,Ssex,Sdept),(Sdept,Deptname),(Sno,Cno,Grade)}

算法6.4 转换为3NF既有无损连接性又保持函数依赖的分解

- 把R分解成3NF，保持FD
- 1.分析 ρ 是否无损。
- 2. 若是则转4。
- 3. ρ = ρ ∪ {X}，X是R的码。
- 4. 输出 ρ 。

- 例：
  S(Sno,Sname,Sage,Ssex,Sdept,Deptname,Cno,Grade)

  F={Sno->Sname,Sno->Sage,Sno->Ssex,Sno->Sdept,Sdept->Deptname,(Sno,Cno)->Grade}

ρ={(Sno, Sname, Sage, Ssex, Sdept), (Sdept, Deptname), (Sno, Cno, Grade)}

算法6.5 转换为BCNF的无损连接分解（分解法）

方法：

1. 置初值 ρ ={R}。
2. 如果 ρ 中所有关系模式已是BCNF转4。否则在 ρ 中找出非BCNF的关系模式S，在S中必有X-〉A，X不包含S的码，A也不在X中。在这种情况下，S中一定有A以外的属性也不包括在X中，用S1，S2代替S，其中S1有A和X组成，S2由S中除A以外的属性组成。
3. 在 ρ 中以S1，S2代替S，转2。
4. 停止分解，输出 ρ 。

设R(Sno,Cno,Grade,TName,Dept)

F={(Sno,Cno)->Grade,Cno->TName,TName->\->Cno,TName->Dept}

转换为BCNF的无损连接分解。

$\rho = \{R1(Sno, Cno, Grade), R2(TName, Dept), R3(Cno, TName)\}$

转换为BCNF的无损连接分解：

F={(Sno,Cno)->Grade,Cno->TName,TName-\->Cno,TName->Dept}

1.不属于BCNF(BCNF左侧都为码，且(Sno,Cno)为码)

2.找到F中右侧不为码的函数依赖Tname->Dept，形成新的关系模式R2(Tname,Dept)，一定为BCNF。

3.R中的属性除Dept，剩余属性形成新的关系模式Rm(Sno,Cno,Grade,Tname),分析是不是BCNF

F’={(Sno,Cno)->Grade，Cno->Tname}码为(Sno,Cno)不是BCNF

4.针对Rm再次分解，找到F’中右侧不为码的函数依赖 Cno->Tname，形成新的关系模式R3(Cno,Tname)，一定为BCNF。

5.Rm中的属性除Tname，剩余属性形成新的关系模式 R1(Sno,Cno,Grade)，(Sno,Cno,Grade) 为BCNF

所以转换为BCNF的无损连接分解

ρ={R1(Sno,Cno,Grade),R2(TName,Dept),R3(Cno,TName)}

但ρ不一定保持函数依赖

解法并不唯一

若先找到F中右侧不为码的函数依赖Cno->Tname，分解完 R2(Cno,Tname)和Rm(Sno,Cno,Grade,Dept)

F’={(Sno,Cno)->Grade,Cno->Dept}

其中Cno->Dept为传递函数依赖

针对Rm再次分解,有

R3(Cno,Dept)和R1(Sno,Cno,Grade)

所以转换为BCNF的无损连接分解

ρ={R1(Sno,Cno,Grade), R2(Cno,TName),R3(Cno,Dept)}

小结(续)

- 规范化理论为数据库设计提供了理论的指南和工具
  - 也仅仅是指南和工具
- 并不是规范化程度越高，模式就越好
  - 必须结合应用环境和现实世界的具体情况合理地选择数据库模式

本章重点和难点

- 判断关系模式是第几范式？
  - 选择、判断候选码
  - 熟记范式要求，可以正确判断
- 判断分解关系模式的性质
  - 无损连结性
  - 保持函数依赖性
- 进行分解关系模式
- 求解极小函数依赖集
- 属性集在函数依赖集上的闭包

判断关系模式的范式

1.R(X,Y,Z) F={XY->Z}
码XY，　　属于BCNF
2.R(X,Y,Z) F={Y->Z,XZ->Y}
码XY和XZ，　属于3NF
3. R(X,Y,Z) F={Y->Z,Y->X,X->YZ}
码X和Y，　　属于BCNF
4. R(X,Y,Z) F={X->Y,X->Z}
码X，　属于BCNF
5. R(W,X,Y,Z) F={X->Z,WX->Y}
码WX，属于1NF

判断分解是否无损和保持函数依赖

1.R(A,B,C),F={A->B}　$\rho=\{AB, BC\}$

否，是

2.R(A,B,C),F={A->B,B->C}　$\rho=\{AC, BC\}$

否，否

3.R(A,B,C),F={A->C,B->C}　$\rho=\{AB, AC\}$

是，否

- 设R(A,B,C,D,E),F={A->BC,CD->E,B->D,E->A}
- 求B^+；R的所有候选码。
- B^+ =BD；
- A，BC，CD，E

- 设R(A,B,C,D,E),F={A->D,E->D,D->B,BC->D,CD->A}
- 求：R的候选码； ρ={AB，AE，CE，BCD，AC}是否为无损连接

- CE，是

- 设R(A,B,C,D,E,G),F={E->D,C->B,CE->G,B->A}
- ρ={DE，BC，CEG，AB}
- 求：是否为无损连接，是否保持函数依赖性

- 是，是

设有R(A,B,C,D) ,F={A->C,C->A,B->AC,D->AC}

1. R的候选码

2.将R分解成BCNF且保持无损连接

3.将R分解成3NF且无损保持函数依赖

BD； ρ={AC，AB，BD}； ρ={AC，BC，CD，BD}

- 设R(A,B,C,D,E),F={A->C,C->D,B->C,DE->C,CE->A}
- 求：R的候选码； ρ={AD，AB，BC，CDE，AE}是否无损连接；将R分解BCNF且无损。

- BE； 否； ρ={AC，BD，ABE}