数据库系统概论

An Introduction to Database System

第六章 关系数据理论

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 ）

  简化为一个二元组：

  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 ）、课程号（ Cno ）
成绩（ Grade ）

单一的关系模式 ： Student <U 、 F>

U ＝ { Sno, Sdept, Mname, Cno, 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. 一个系只有一名系主任；
3. 一个学生可以选修多门课程， 每门课程有若干学生选修；

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

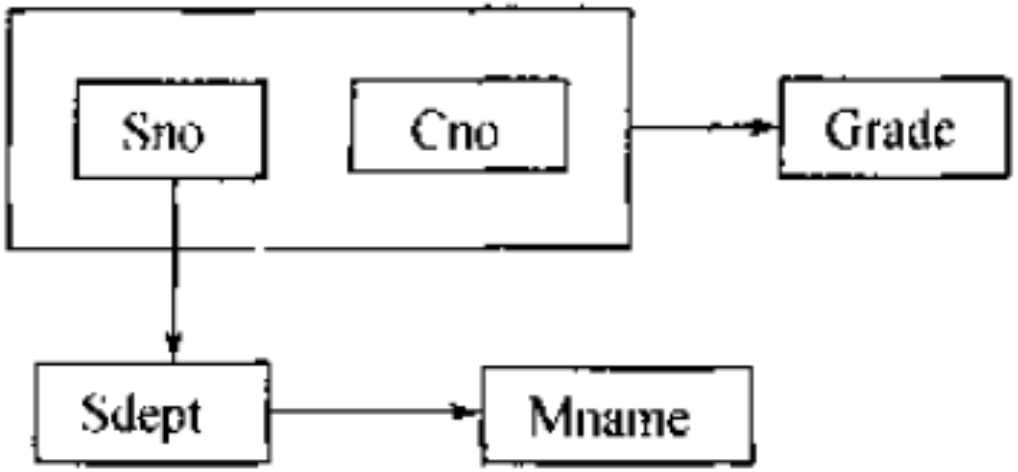


图 6.1 Student 上的一组函数依赖

关系模式 Student<U, F> 中存在的问题

表 6.1 Student 表

Sno	Sdept	Mname	Cno	Grade
S1	计算机系	张明	C1	95
S2	计算机系	张明	C1	90
S3	计算机系	张明	C1	88
S4	计算机系	张明	C1	70
S5	计算机系	张明	C1	78
⋮	⋮	⋮	⋮	⋮

1. 数据冗余太大

   – 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

表 6.1 Student 表

Sno	Sdept	Mname	Cno	Grade
S1	计算机系	张明	C1	95
S2	计算机系	张明	C1	90
S3	计算机系	张明	C1	88
S4	计算机系	张明	C1	70
S5	计算机系	张明	C1	78
⋮	⋮	⋮	⋮	⋮

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 $X \rightarrow Y$ 。

X 称为这个函数依赖的决定属性集 (Determinant) 。

记作 Y=f(x)

一、函数依赖

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

但 Ssex →Sage

一、函数依赖

1. 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

2. 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。

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

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' ↛ 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 $X \xrightarrow{P} Y$。

三、完全函数依赖与部分函数依赖

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 是完全函数依赖

在例 U ＝｛ Sno, Sdept, Mname, Cno, Grade ｝

(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，而 Sno 是（ Sno ， Cno ）的真子集。

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 $Y \subseteq X$，　$Y \not\to X$，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, Mname) 中，有：

Sno → Sdept，　Sdept → Mname

Mname 传递函数依赖于 Sno，记为：

$\text{Sno} \xrightarrow{\text{传递}} \text{Mname}$

→

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)

6.2.3 范式

- 各种范式之间存在联系：

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

- 某一关系模式 R 为第 n 范式，可简记为 $R \in nNF$ 。

第一范式： 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。

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

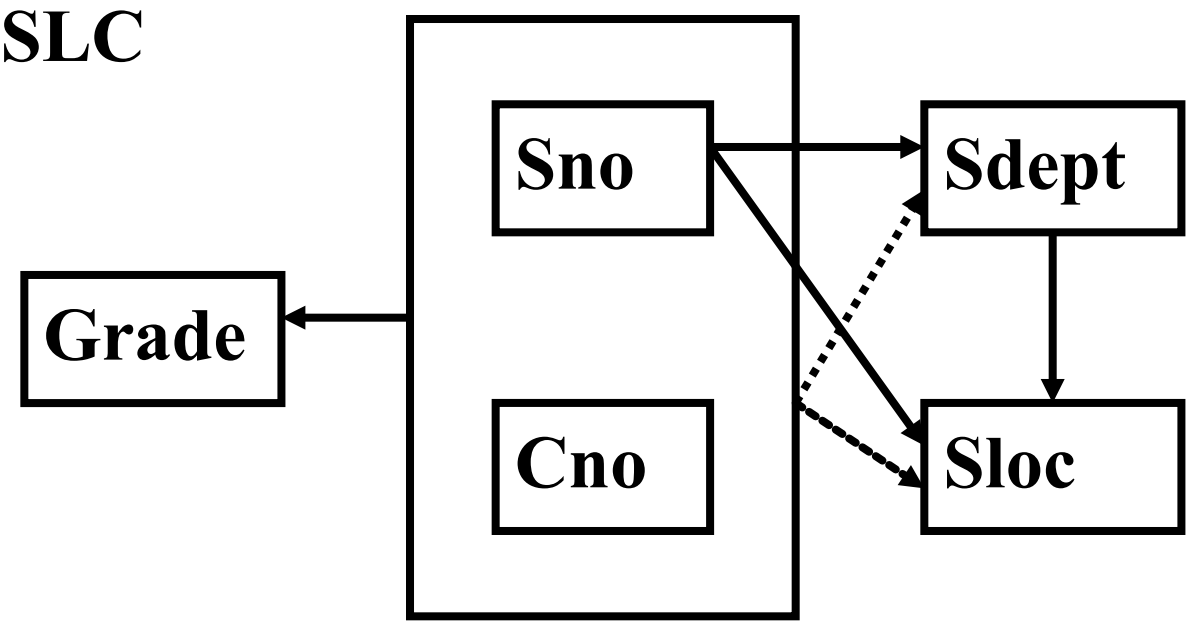
  (Sno, Cno) $\xrightarrow{P}$ Sdept

  Sno → Sloc

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

  Sdept → Sloc

  →



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

关系模式 SLC(Sno, Cno , Sdept, Sloc, Grade)

(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

关系模式 SLC(Sno, Sdept, Sloc, Cno, Grade)

(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。

SLC 不是一个好的关系模式

- 原因

  Sdept 、 Sloc 部分函数依赖于码（见后图）

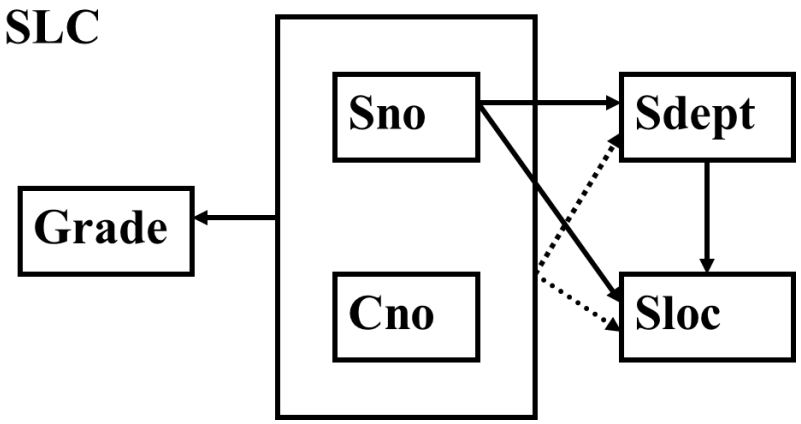
- 解决方法

  将 SLC (Sno, Sdept, Sloc, Cno, Grade) 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 ）

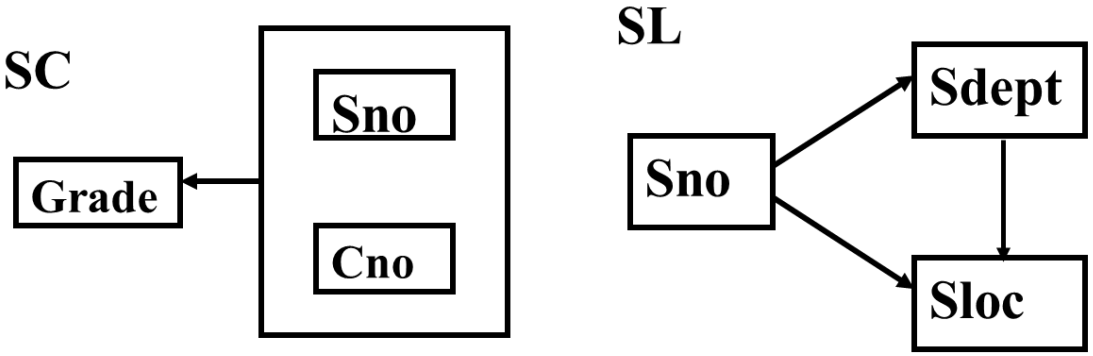
SLC 不是一个好的关系模式



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)

函数依赖图：

转化为：



第二范式： 2NF

- **2NF 的定义**

定义 6.6 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： 2NF

- 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。

SC （ Sno， Cno， Grade ）

SL （ Sno， Sdept， Sloc ）

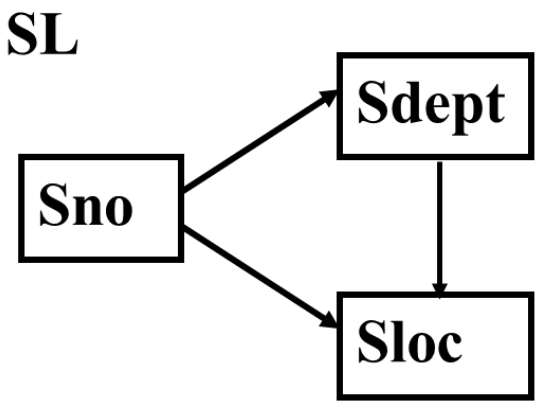
例：新建楼系没学生时？删学生可能会删系及住所？学生转系必须要多处改住所。。。

2NF 关系模式 SL(Sno, Sdept, Sloc) 中

- 函数依赖：

  Sno→Sdept ， Sdept→Sloc ， Sno→Sloc

  函数依赖图：



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。

第三范式： 3NF

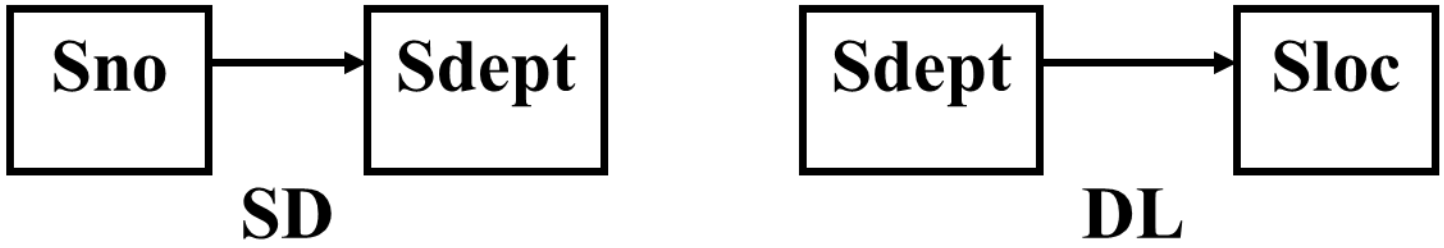
- 解决方法

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 。



不存在传递依赖

第三范式： 3NF

- **3NF 的定义**

  定义 6.7　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（$Z \subseteq Y$），使得 $X \rightarrow Y$，$Y \rightarrow X$，$Y \rightarrow Z$，成立，则称 $R<U, F> \in 3NF$。

（即：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。）

例，　SL(Sno, Sdept, Sloc) ∈ 2NF

但　SL(Sno, Sdept, Sloc) ∈ 3NF，分解后

SD（Sno，　Sdept）　∈　3NF

DL（Sdept，　Sloc）∈　3NF

第三范式： 3NF

1）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。

2）如果 R∈3NF，则 R 也是 2NF 。

3）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。

4）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。

6.2.9 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

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

规范化的基本思想

- 消除不合适的数据依赖
- 的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

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

习题

- 已知关系模式

销售（项目号，零件号，零件价格，零件数量，供应商，供应商所在地区）

其中，项目号→供应商。其它依赖根据现实关系自行分析，将该关系模式转化为 3NF 。

- 分析：有如下函数依赖：

（项目号，零件号）→零件数量

零件号→零件价格

项目号→供应商

供应商→供应商所在地区

- 销售（项目号，零件号，零件数量）
- 零件（零件号，零件价格）
- 项目（项目号，供应商）
- 供应商（供应商，供应商所在地区）

- 设关系模式 R(ABCD)， F 是 R 上成立的函数依赖集， F={A→C，C→B}，则相对于 F，写出关系模式 R 的主关键字。
- 解： R(ABCD)，F={A→C，C→B}
- AD→A， AD→D
- 由 A→C，C→B 知 A→B
- 所以 AD→B, AD→C
- AD→ABCD
- 主键是 AD

- 设关系模式 R(ABCD) , F={ AB→CD ， A→D }
- (1) 试说明 R 不是 2NF 模式的理由。
- (2) 试把 R 分解成 2NF 模式集。
- 解： R(ABCD) , F={ AB→CD ， A→D }
- (1)R 的候选键是 AB
- 由于 A→D，存在非主属性对候选键的部分依赖 , R 不是 2NF
- (2) R 应分解为 ρ={AD ， ABC }
- ρ 是 2 NF 模式

- 设关系模式 R(ABC) , F={ C→B ， B→A }
- (1) 试说明 R 不是 3NF 模式的理由。
- (2) 试把 R 分解成 3NF 模式集。
- 解： R(ABC) , F={ C→B ， B→A }
- (1) 主键是 C，C→B ， B→A 可知 C→A
- 且 A 是非主属性，存在传递函数依赖，所以 R 不是 3NF
- (2)R 应分解为 ρ={CB ， BA } , ρ 是 3 NF 模式

设有关系模式R(A，B，C，D，E，F),其函数依赖集为：F={E→D，C→B，CE→F，B→A}。

请回答如下问题：

(1)指出R的所有候选码并说明原因；

(2)R最高属于第几范式，为什么？

(3)分解R为3NF。

可知 A 、 B 、 D 、 F 四个属性均不是决定因素，所以只有 C 和 E 有可能构成该关系模式的主键，而 C 、 E 之间没有函数依赖关系，且根据已知的函数依赖可知， CE→ABCDEF ，所以 R 的主键是 CE 。

CE →B 但 C →B

CE →D 但 E →D

CE → A 但 C → B 且 B → A （ C → A ）（部分函数依赖）

则有： R1 （ C ， A ， B ） R2 （ E ， D ）

R3 （ C ， E ， F ）

举例：学生成绩登记表

学号	姓名	性别	专业	年级	课程成绩						
					课号	课名	学时	学分	教师	工资号	成绩
S1	张三	男	CS	98	C1	DB	60	3	赵	M1	90
					C2	DS	60	3	钱	M9	70
					C3	OS	80	4	孙	M4	85
					C4	MA	120	6	李	M7	90
					C5	PH	90	5	周	M2	75
S2	李四	女	CS	99	C1	DB	60	3	赵	M1	86

- 消去可划分的属性：课程成绩
  - 学生(学号,姓名,性别,专业,年级,课号,课名,学分,学时,教师,工资号,成绩)
  - 关键字(学号,课号)
  - $\in$1NF

举例：学生成绩登记表

学号	姓名	性别	专业	年级	课程成绩						
					课号	课名	学时	学分	教师	工资号	成绩
S1	张三	男	CS	98	C1	DB	60	3	赵	M1	90
					C2	DS	60	3	钱	M9	70
					C3	OS	80	4	孙	M4	85
					C4	MA	120	6	李	M7	90
					C5	PH	90	5	周	M2	75
S2	李四	女	CS	99	C1	DB	60	3	赵	M1	86

- 消去部分函数依赖
  - 存在的部分依赖：
    - (学号,课号) p→ (姓名,性别,专业,年级)
    - (学号,课号) p→ (课名,学分,学时,工资号,教师)
  - 消去部分依赖
    - (学号)→(姓名,性别,专业,年级)
    - (课号)→(课名,学分,学时,工资号,教师)
    - (学号,课号)→成绩
  - 投影成三个子关系模式
    - 学生(学号,姓名,性别,专业,年级)
    - 课程(课号,课名,学分,学时,工资号,教师)
    - 成绩(学号,课号,成绩)
  - ∈2NF

举例：学生成绩登记表

学号	姓名	性别	专业	年级	课程成绩						
					课号	课名	学时	学分	教师	工资号	成绩
S1	张三	男	CS	98	C1	DB	60	3	赵	M1	90
					C2	DS	60	3	钱	M9	70
					C3	OS	80	4	孙	M4	85
					C4	MA	120	6	李	M7	90
					C5	PH	90	5	周	M2	75
S2	李四	女	CS	99	C1	DB	60	3	赵	M1	86

- 消去传递函数依赖
  - 存在的传递依赖
    - 课号→工资号
    - 工资号→教师
    - 课号 t→ 教师
  - 消去传递依赖
    - (课号)→(课名,学分,学时,工资号)
    - (工资号→教师)
  - 投影成两个子关系模式
    - 课程(课号,课名,学分,学时,工资号)
    - 教师(工资号,教师)
  - ∈3NF

举例：学生成绩登记表

学号	姓名	性别	专业	年级	课程成绩						
					课号	课名	学时	学分	教师	工资号	成绩
S1	张三	男	CS	98	C1	DB	60	3	赵	M1	90
					C2	DS	60	3	钱	M9	70
					C3	OS	80	4	孙	M4	85
					C4	MA	120	6	李	M7	90
					C5	PH	90	5	周	M2	75
S2	李四	女	CS	99	C1	DB	60	3	赵	M1	86

- 最后投影结果
  - 学生(学号,姓名,性别,专业,年级)
  - 课程(课号,课名,学分,学时,师号)
  - 教师(师号,教师)
  - 成绩(学号,课号,成绩)