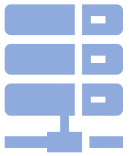
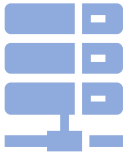
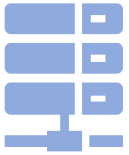


第十一章 关系数据理论



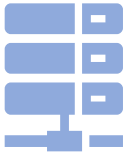
数据库设计

- 如何使用关系模型设计关系数据库？也就是面对一个现实问题，如何选择一个比较好的关系模式的集合，每个关系又应该由哪些属性组成？这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
- 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
- 有关数据库设计的全过程将在下一章详细讨论



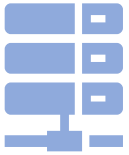
规范化理论

- 规范化理论由关系数据库的创始人E.F.Codd提出
- 在该理论出现以前，层次和网状数据库的设计只是遵循其模型本身固有的原则，而无具体的理论依据可言，因而带有盲目性，可能在以后的运行和使用中发生许多预想不到的问题
- 在关系数据库系统中，关系模型包括一组关系模式，各个关系不是完全孤立的，数据库的设计较层次和网状模型更为重要。



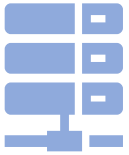
规范化理论

- 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，这些工作决定了整个系统运行的效率，也是系统成败的关键所在，所以必须在关系数据库的规范化理论的指导下逐步完成



规范化理论

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



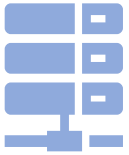
规范化问题的提出

- 关系模式的存储异常问题
  - 数据库的逻辑设计为什么要遵循一定的规范化理论?
  - 什么是好的关系模式?
  - 某些不好的关系模式可能导致哪些问题?
- 下面通过例子进行分析

要求设计教学管理数据库，设计出来的关系模式SCD如下：

SCD(SNO,SN,AGE,DEPT,MN,CNO,SCORE)

其中，SNO表示学生学号，SN表示学生姓名，AGE表示学生年龄，DEPT表示学生所在的系别，MN表示系主任姓名，CNO表示课程号，SCORE表示成绩。



规范化问题的提出

- 根据实际情况，有如下语义规定：
  - 一个系有若干个学生，但一个学生只属于一个系
  - 一个系只有一名系主任，但一个系主任可以同时兼几个系的系主任
  - 一个学生可以选修多门功课，每门课程可有若干学生选修
  - 每个学生学习课程有一个成绩。

规范化问题的提出

- SCD关系模式实例的一个示例：

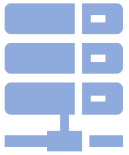
SCD

SNO	SN	AGE	DEPT	MN	CNO	SCORE
S1	赵亦	17	计算机	刘伟	C1	90
S1	赵亦	17	计算机	刘伟	C2	85
S2	钱尔	18	信息	王平	C5	57
S2	钱尔	18	信息	王平	C6	80
S2	钱尔	18	信息	王平	C7	70
S2	钱尔	18	信息	王平	C5	70
S3	孙珊	20	信息	王平	C1	0
S3	孙珊	20	信息	王平	C2	70
S3	孙珊	20	信息	王平	C4	85
S4	李思	男	自动化	刘伟	C1	93

规范化问题的提出

- 根据用户需求描述，可以了解到(SNO,CNO)属性的组合能唯一标识一个元组，所以 确定(SNO,CNO)为该关系模式的主键
- 使用设计完成的关系模式SCD进行数据操作，发现如下的问题：
  - 数据冗余

    例如每个系名和系主任的名字存储的次数等于该系的学生人数乘以每个学生选修的课程门数，同时学生的姓名、年龄也都要重复存储多次，数据的冗余度很大，浪费了存储空间

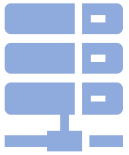


规范化问题的提出

- 使用设计完成的关系模式SCD进行数据操作，发现如下的问题：
  - 插入异常

    如果某个新系没有招生，尚无学生时，则系名和系主任的信息无法插入到数据库中。

    因为在这个关系模式中，(SNO,CNO)是主键。根据关系的实体完整性约束，主键的值不能为空，而这时没有学生，SNO和CNO均无值，因此不能进行插入操作。另外，当某个学生尚未选课，即CNO未知，实体完整性约束还规定，主键的值不能部分为空，同样不能进行插入操作。



规范化问题的提出

- 使用设计完成的关系模式SCD进行数据操作，发现如下的问题：
  - 删除异常。

某系学生全部毕业而没有招生时，删除全部学生的记录则系名、系主任也随之删除，而这个系依然存在，在数据库中却无法找到该系的信息。

另外，如果某个学生不再选修C1课程，本应该只删去C1，但C1是主关系键的一部分，为保证实体完整性，必须将整个元组一起删掉，这样，有关该学生的其它信息也随之丢失。

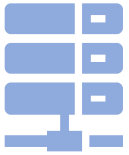
规范化问题的提出

- 使用设计完成的关系模式SCD进行数据操作，发现如下的问题：
  - 更新异常

    如果学生改名，则该学生的所有记录都要逐一修改SN；

    又如某系更换系主任，则属于该系的学生记录都要修改MN的内容，稍有不慎，就有可能漏改某些记录，这就会造成数据的不一致性，破坏了数据的完整性。

由于存在以上问题，我们说，SCD是一个不好的关系模式。产生上述问题的原因，直观地说，是因为关系中“包罗万象”，内容庞杂且互相影响。



规范化问题的提出

- 那么，什么样的才是一个好的关系模式呢?

- 我们把关系模式SCD分解为下面三个结构简单的关系模式:
  - 学生关系S(SNO,SN,AGE,DEPT)
  - 选课关系SC(SNO,CNO,SCORE)
  - 系关系D(DEPT,MN)

规范化问题的提出

- 好的关系模式

S

SNO	SN	AGE	DEPT
S1	赵亦	17	计算机
S2	钱尔	18	信息
S3	孙珊	20	信息
S4	李思	21	自动化

D

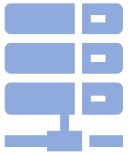
DEPT	MN
计算机	刘伟
信息	王平
自动化	刘伟

SC

SNO	CNO	SCORE
S1	C1	90
S1	C2	85
S2	C5	57
S2	C6	80
S2	C7	
S2	C5	70
S3	C1	0
S3	C2	70
S3	C4	85
S4	C1	93

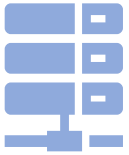
规范化问题的提出

- 在以上三个关系模式中，实现了信息的某种程度的分离，
  - S中存储学生基本信息，与所选课程及系主任无关；
  - D中存储系的有关信息，与学生无关；
  - SC中存储学生选课的信息，而与学生及系的有关信息无关。



规范化问题的提出

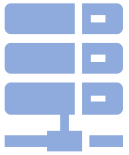
- 与SCD相比，分解为三个关系模式后，数据的冗余度明显降低。当新插入一个系时，只要在关系D中添加一条记录。
- 当某个学生尚未选课，只要在关系S中添加一条学生记录，而与选课关系无关，这就避免了插入异常。
- 当一个系的学生全部毕业时，只需在S中删除该系的全部学生记录，而关系D中有关该系的信息仍然保留，从而不会引起删除异常。
- 同时，由于数据冗余度的降低，数据没有重复存储，也不会引起更新异常。



规范化问题的提出

- 经过上述分析，我们说分解后的关系模式是一个好的关系数据库模式。

- 从而得出结论，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。



规范化问题的提出

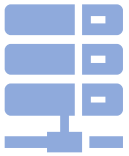
- 如何按照一定的规范设计关系模式，将结构复杂的关系分解成结构简单的关系，从而把不好的关系数据库模式转变为好的关系数据库模式，这就是关系的规范化。
- 规范化又可以根据不同的要求而分成若干级别。
- 我们要设计的关系模式中的各属性是相互依赖、相互制约的，这样才构成了一个结构严谨的整体。
- 因此在设计关模式时，必须从语义上分析这些依赖关系。
- 数据库模式的好坏和关系中各属性间的依赖关系有关，因此，我们先讨论属性间的依赖关系，然后再讨论关系规范化理论。

第十一章 关系数据理论

1. 函数依赖
2. 范式
3. 模式分解

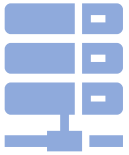
函数依赖

- 函数依赖的定义及性质
  - 关系模式中的各属性之间相互依赖、相互制约的联系称为数据依赖。
- 数据依赖一般分为函数依赖、多值依赖和连接依赖。
- 其中,函数依赖是最重要的数据依赖。



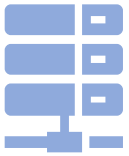
函数依赖

- 函数依赖（Functional Dependency）是关系模式中属性之间的一种逻辑依赖关系。
  - 例如在上一节介绍的关系模式SCD中，SNO与SN、AGE、DEPT之间都有一种依赖关系。
  - 由于一个SNO只对应一个学生，而一个学生只能属于一个系，所以当SNO的值确定之后，SN，AGE，DEPT的值也随之被唯一的确定了
  - 这类似于变量之间的单值函数关系。设单值函数Y=F(X)，自变量X的值可以决定一个唯一的函数值Y。
  - 在这里，我们说SNO决定函数（SN，AGE，DEPT），或者说（SN，AGE，DEPT）函数依赖于SNO。



函数依赖

- 下面给函数依赖的形式化定义。
  - 设关系模式R(U，F)，U是属性全集，F是U上的函数依赖集，X和Y是U的子集，如果对于R(U)的任意一个可能的关系r，对于X的每一个具体值，Y都有唯一的具体值与之对应，则称X决定函数Y，或Y函数依赖于X，记作X→Y。我们称X为决定因素，Y为依赖因素。当Y不函数依赖于X时，记作：X ↛ Y。当X→Y且Y→X时，则记作：X↔Y。



函数依赖

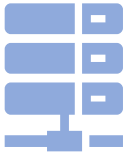
- 对于关系模式SCD

  U={SNO,SN,AGE,DEPT,MN,CNO,SCORE}

- 很显然：SNO→SN，SNO→AGE，SNO→DEPT
- 一个SNO有多个SCORE的值与其对应，因此SCORE不能唯一地确定，即SCORE不能函数依赖于SNO，所以有：SNO ↛ SCORE。
- 但是SCORE可以被（SNO，CNO）唯一地确定。所以可表示为：（SNO，CNO）→SCORE。
- （SNO，CNO,AGE）→SCORE

函数依赖

- 有关函数依赖的几点说明
  - 1. 平凡的函数依赖与非平凡的函数依赖
    - ✓ 当属性集Y是属性集X的子集时，则必然存在着函数依赖X→Y，这种类型的函数依赖称为平凡(Trivial)的函数依赖。 （SNO，CNO）→SNO
    - ✓ 如果Y不是X的子集，则称X→Y为非平凡的函数依赖。
    - ✓ 若不特别声明，我们讨论的都是非平凡的函数依赖。

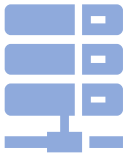


函数依赖

- 有关函数依赖的几点说明
  - 2.函数依赖是语义范畴的概念
    - ✓ 我们只能根据语义来确定一个函数依赖，而不能按照其形式化定义来证明一个函数依赖是否成立。
    - ✓ 例如，对于关系模式S，当学生不存在重名的情况下，可以得到

      SN→AGE

      SN→DEPT
    - ✓ 这种函数依赖关系，必须是在没有重名的学生条件下才成立的，否则就不存在函数依赖了。
    - ✓ 所以函数依赖反映了一种语义完整性约束。

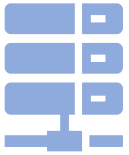


函数依赖

- 有关函数依赖的几点说明
  - 3. 函数依赖关系的存在与时间无关
    - ✓ 因为函数依赖是指关系中的所有元组应该满足的约束条件，而不是指关系中某个或某些元组所满足的约束条件。
    - ✓ 当关系中的元组增加、删除或更新后都不能破坏这种函数依赖。
    - ✓ 因此，必须根据语义来确定属性之间的函数依赖，而不能单凭某一时刻关系中的实际数据值来判断。

函数依赖

- 有关函数依赖的几点说明
  - 3. 函数依赖关系的存在与时间无关
    - ✓ 例如，对于关系模式S，假设没有给出无重名的学生这种语义规定，则即使当前关系中没有重名的记录，也只能存在函数依赖SNO→SN，而不能存在函数依赖SN→SNO，因为如果新增加一个重名的学生，函数依赖SN→SNO必然不成立。
    - ✓ 所以函数依赖关系的存在与时间无关，而只与数据之间的语义规定有关。

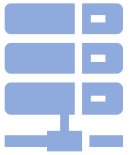


函数依赖

- 有关函数依赖的几点说明
  - 4. 函数依赖可以保证关系分解的无损连接性
    - ✓ 设R（X，Y，Z），X，Y，Z为不相交的属性集合，如果X→Y或X→Z，则有R(X，Y，Z) = R1[X，Y] ⋈ R2[X，Z]
    - ✓ 其中，R1[X，Y]表示关系R在属性（X，Y）上的投影
    - ✓ 即R等于其投影在X上的自然连接，这样便保证了关系R分解后不会丢失原有的信息，称作关系分解的无损连接性。
    - ✓ 基于上述原则的分解方法称为投影分解

函数依赖

- 有关函数依赖的几点说明
  - 4. 函数依赖可以保证关系分解的无损连接性
    - ✓ 例如，对于关系模式SCD，有SNO→（SN，AGE，DEPT，MN），SCD（SNO，SN，AGE，DEPT，MN，CNO，SCORE）=A[SNO，SN，AGE，DEPT，MN]⋈B[SNO，CNO，SCORE]，也就是说，用其投影在SNO上的自然连接可复原关系模式SCD
    - ✓ 这一性质非常重要，在后面的关系规范化中要用到。



函数依赖

- 函数依赖的分类
  - 完全函数依赖与部分函数依赖
    - ✓ 设关系模式R(U)，U是属性全集，X和Y是U的子集
    - ✓ 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对X完全函数依赖（Full Functional Dependency），记作 $X \xrightarrow{f} Y$。
    - ✓ 如果对X的某个真子集X′，有X′→Y，则称Y对部分函数依赖（Partial Functional Dependency），记作$X \xrightarrow{p} Y$。
      - ✓ 例如，在关系模式SCD中，因为SNO ↛ SCORE，且
      - ✓ CNO ↛ SCORE，所以有：（SNO，CNO）$\xrightarrow{f}$ SCORE 。
      - ✓ 而SNO→AGE，所以（SNO，CNO）$\xrightarrow{p}$ AGE

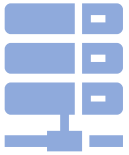
函数依赖

- 函数依赖的分类
  - 完全函数依赖与部分函数依赖
    - ✓ 由上述定义可知：

      只有当决定因素是组合属性时，讨论部分函数依赖才有意义，

      当决定因素是单属性时，只能是完全函数依赖。

      例如，在关系模式S（SNO，SN，AGE，DEPT），决定因素为单属性SNO，有SNO→（SN，AGE，DEPT），不存在部分函数依赖。



函数依赖

- 函数依赖的分类
  - 完全函数依赖与部分函数依赖
    - ✓ 出版（ISBN号，书名，书价，印刷厂名，厂址）
    - ✓ 销售（商品编号，售货员工号，价格，购买数量，购买时间，顾客编号，顾客地址）
    - ✓ 出版（ISBN号，书名，书价，出版社名，出版社地址）

函数依赖

- 函数依赖的分类
  - 传递函数依赖
    - ✓ 设有关系模式R（U），U是属性全集，X，Y，Z是U的子集，
    - ✓ 若X→Y，但Y ↛ X，而Y→Z（Y∉X，Z∉Y），则称Z对X传递函数依赖（Transitive Functional Dependency），记作：$X \xrightarrow{t} Z$。
    - ✓ 如果Y→X，则X↔Y，这时称Z对X直接函数依赖，而不是传递函数依赖。

函数依赖

- 函数依赖的分类
  - 传递函数依赖
    - ✓ 例如，在关系模式SCD中，SNO→DEPTN，但DEPTN ↛ SNO，而DEPTN→MN，则有SNO $\xrightarrow{t}$ MN。当学生不存在重名的情况下，有SNO→SN，SN→SNO，SNO ↔ SN，SN→DEPTN，这时DEPTN对SNO是直接函数依赖，而不是传递函数依赖。
    - ✓ 生产（产品编号，生产部门，部门地址，质检员，质检员性别）
  - 综上所述，函数依赖分为完全函数依赖、部分函数依赖和传递函数依赖三类，它们是规范化理论的依据和规范化程度的准则

函数依赖

- 用函数依赖定义候选键
  - 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的一个候选键（Candidate Key）。
  - 若关系模式R有多个候选键，则选定其中的一个做为主键（Primary key）。
  - 若k是R的一个候选键，并且S⊃K，则称S是R的一个超键 (Super Key)
  - 在任一候选键中出现过的属性称为主属性；主属性之外的称为非主属性

练习：

学生（学号，姓名，性别，系名，系主任）

产品（产品编号，产品名称，产品类别，价格，生产厂家，厂址）

产品（产品名称，产品类别，价格，生产厂家，厂址）

函数依赖

用函数依赖定义外键

- 关系模式 R 中属性或属性组 X 并非 R 的候选键，但 X 是另一个关系模式的候选键，则称 X 是R 的外键（Foreign key）

练习：

学生（学号，姓名，性别，系名）

系（系号，系名，系主任）

产品（产品名称，产品类别，价格，生产地点（厂址））

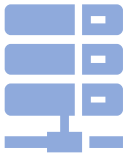
生产厂家（工厂编号，厂家名称，厂址，法人代表）

出版（ISBN号，出版社名称，出版日期）

书（ISBN号，书名，价格）

第十一章 关系数据理论

1. 函数依赖
2. 范式
3. 模式分解



范式

- 规范化的基本思想是消除关系模式中的数据冗余，消除数据依赖中的不合适的部分，解决数据插入、删除时发生的异常现象。
- 这就要求关系数据库设计出来的关系模式要满足一定的条件。
- 我们把关系数据库的规范化过程中为不同程度的规范化要求设立的不同标准称为范式（Normal Form）。
- 由于规范化的程度不同，就产生了不同的范式。
  - 满足最基本规范化要求的关系模式叫第一范式，
  - 在第一范式中进一步满足一些要求为第二范式，
  - 以此类推就产生了第三范式等概念。
- 每种范式都规定了一些限制约束条件。

范式

- 范式的概念最早由E.F.Codd提出。
- 从1971年起，Codd相继提出了关系的三级规范化形式，即第一范式（1NF）、第二范式（2NF）、第三范式（3NF）。
- 1974年，Codd和Boyce以共同提出了一个新的范式的概念，即Boyce-Codd范式，简称BC范式。
- 1976年Fagin提出了第四范式，
- 后来又有人定义了第五范式。
- 至此在关系数据库规范中建立了一个范式系列：1NF,2NF,3NF,BCNF,4NF,5NF,一级比一级有更严格的要求。

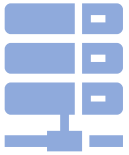
范式

- 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。

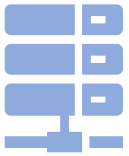
- 下面逐一介绍各范式定义和特点



1NF

- 第一范式（First Normal Form）是最基本的规范形式，即关系中每个属性都是不可再分的简单项。
- 定义 如果关系模式R，其所有的属性均为简单属性，即每个属性域都是不可再分的，则称R属于第一范式，简称1NF，记作$R \in 1NF$。

- 第一范式排除了多值属性、组合属性



1NF

- 一个关系模式仅仅属于第一范式是不够的。在前面给出的关系模式SCD属于第一范式，但其具有大量的数据冗余，具有插入异常、删除异常、更新异常等弊端。
- 为什么会出现这些问题呢?
- 分析一下SCD中的函数依赖关系，它的关系键是（SNO，CNO）的属性组合，所以有如下函数依赖：

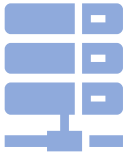
$$(SNO, CNO) \xrightarrow{f} SCORE$$

$$SNO \rightarrow SN, \quad (SNO, CNO) \xrightarrow{p} SN$$

$$SNO \rightarrow AGE, \quad (SNO, CNO) \xrightarrow{p} AGE$$

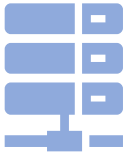
$$SNO \rightarrow DEPT, \quad (SNO, CNO) \xrightarrow{p} DEPT$$

$$SNO \xrightarrow{t} MN, \quad (SNO, CNO) \xrightarrow{p} MN$$



1NF

- 由此可见，在SCD中，既存在完全函数依赖，又存在部分函数依赖和传递函数依赖。
- 由于关系中存在着复杂的函数依赖，才导致数据操作中出现了种弊端。
- 克服这些弊端的方法是用投影运算将关系分解，去掉过于复杂的函数依赖关系，向更高一级的范式进行转换。



2NF

- 如果关系模式R ∈ 1NF，且每个非主属性都完全函数依赖于R的每个关系键，则称R属于第二范式（Second Normal Form），简称2NF，记作R ∈ 2NF。

- 在关系模式SCD中，SNO，CNO为主属性，AGE，DEPT，MN，MN，SCORE均为非主属性，经前述分析，存在非主属性对关系键的部分函数依赖，所以SCD不是2NF。

2NF

- 将SCD分解成三个关系模式S，D，SC：
  - 学生关系S(SNO,SN,AGE,DEPT,MN)
  - 选课关系SC(SNO,CNO,SCORE)
- 其中S的关系键为SNO，D的关系键为DEPT，都是单属性，不可能存在部分函数依赖。而对于SC，(SNO，CNO)　　SCORE。所以SCD分解后，消除了非主属性对关系键的部分函数依赖，S，D，SC均属于2NF。

2NF

- 又如关系模式TCS（T，C，S）
  - 一个教师可以讲授多门课程，一门课程可以是多个教师讲授
  - 同样一个学生可以选听多门课程，一门课程可以为多个学生选听
- (T,C,S)三个属性的组合是关系键，T,C,S都是主属性，而无非主属性，所以也就不可能存在非主属性对关系键的部分函数依赖，TCS $\in$ 2NF。

2NF

- 经以上分析，可以得到两个结论：
  - 1．从1NF关系中消除非主属性对关系键的部分函数依赖，则可得到2NF关系。
  - 2．如果R的关系键为单属性，或R的全体属性均为主属性，则R $\in$ 2NF。

2NF

- 2NF规范化
  - 2NF规范化是指把1NF关系模式通过投影分解转换成2NF关系模式的集合。
  - 分解时遵循的基本原则就是“一事一表”，让一个关系只描述一个实体或者实体间的联系。如果多于一个实体或联系，则进行投影分解。

2NF

- 下面对2NF规范化作形式化的描述：
- 设关系模式R（X，Y，Z），R ∈ 1NF，但R不是2NF，其中，X是主属性，Y，Z是非主属性，且存在部分函数依赖，$X \xrightarrow{p} Y$。设X可表示为X_1、X_2，其中$X_1 \xrightarrow{f} Y$。则R（X，Y，Z）可以分解为R[X_1，Y]和R[X，Z]

2NF

- 因为$X_1 \rightarrow Y$，所以R(X，Y，Z)=R[X1，Y] ⋈ R[X1，X2，Z]=R[X1，Y] ⋈ R[X，Z]，即R等于其投影R[X_1，Y]和[X，Z]在X_1上的自然连接，R的分解具有无损连接性。
- 由于$X_1 \xrightarrow{f} Y$，因此R[X1，Y] ∈ 2NF。若R[X，Z]不属于2NF，可以按照上述方法继续进行投影分解，直到将R[X，Z]分解为属于2NF关系的集合，且这种分解必定是有限的

2NF

- 下面以关系模式SCD为例，来说明2NF规范化的过程

  SCD(SNO,SN,AGE,DEPT,MN,CNO,SCORE)

- 由SNO→SN，SNO→AGE，SNO→DEPT，

（SNO，CNO）$\xrightarrow{f}$SCORE，可以判断，关系SCD至少描述了两个实体：

  - 一个为学生实体，属性有SNO、SN、AGE、DEPT、MN；
  - 另一个是学生与课程的联系（选课），属性有SNO、CNO和SCORE。

- 根据投影分解的原则，我们可以将SCD分解成两个关系，如下页所示

2NF

- SD(SNO,SN,AGE,DEPT，MN)，描述学生实体；
- SC(SNO,CNO,SCORE)，描述学生与课程的联系。

- 对于分解后的两个关系SD和SC，主键分别为SNO和（SNO，CNO），非主属性对主键完全函数依赖。因此，SD ∈ 2NF，SC ∈ 2NF，而且前面已经讨论，SCD的这种分解没有丢失任何信息，具有无损连接性。

2NF

■ 关系实例

SD

SNO	SN	AGE	DEPT	MN
S1	赵亦	17	计算机	刘伟
S2	钱尔	18	信息	王平
S3	孙珊	20	信息	王平
S4	李思	21	自动化	刘伟

SC

SNO	CNO	SCORE
S1	C1	90
S1	C2	85
S2	C5	57
S2	C6	80
S2	C7	
S2	C5	70
S3	C1	0
S3	C2	70
S3	C4	85
S4	C1	93

关系SD和SC

2NF

■ 关系实例

与SCD对比

SCD

SNO	SN	AGE	DEPT	MN	CNO	SCORE
S1	赵亦	17	计算机	刘伟	C1	90
S1	赵亦	17	计算机	刘伟	C2	85
S2	钱尔	18	信息	王平	C5	57
S2	钱尔	18	信息	王平	C6	80
S2	钱尔	18	信息	王平	C7	70
S2	钱尔	18	信息	王平	C5	70
S3	孙珊	20	信息	王平	C1	0
S3	孙珊	20	信息	王平	C2	70
S3	孙珊	20	信息	王平	C4	85
S4	李思	男	自动化	刘伟	C1	93

2NF

- 1NF的关系模式经过投影分解转换成2NF后，消除了一些数据冗余。
- 分析图中SD和SC中的数据，可以看出，它们存储的冗余度比关系模式SCD有了较大辐度的降低。
  - ✓ 学生的姓名、年龄不需要重复存储多次。
- 这样便可在一定程度上避免数据更新所造成的数据不一致性的问题。

2NF

- 由于把学生的基本信息与选课信息分开存储，则学生基本信息因没选课而不能插入的问题得到了解决，插入异常现象得到了部分改善。
- 同样，如果某个学生不再选修C1课程，只在选课关系SC中删去该该学生选修C1的记录即可，而SD中有关该学生的其它信息不会受到任何影响，也解决了部分删除异常问题。
- 因此可以说关系模式SD和SC在性能上比SCD有了显著提高

2NF

- 2NF的缺点
- 2NF的关系模式解决了1NF中存在的一些问题，2NF规范化的程度比1NF前进了一步，但2NF的关系模式在进行数据操作时，仍然存在着一些问题：
  - ✓ 1．数据冗余。每个系名和系主任的名字存储的次数等于该系的学生人数。
  - ✓ 2．插入异常。当一个新系没有招生时，有关该系的信息无法插入。
  - ✓ 3．删除异常。某系学生全部毕业而没有招生时，删除全部学生的记录也随之删除了该系的有关信息。
  - ✓ 4．更新异常。更换系主任时，仍需改动较多的学生记录。

2NF

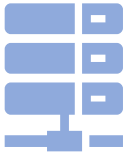
- 2NF的缺点
- 之所以存在这些问题，是由于在SCD中存在着非主属性对主键的传递依赖。
- 分析SCD中的函数依赖关系，SNO→SN，SNO→AGE，SNO→DEPT，DEPT→MN，SNO $\xrightarrow{t}$ MN，非主属性MN对主键SNO传递依赖。
- 为此，对关系模式SCD还需进一步简化，消除这种传递依赖，得到3NF。

3NF

- 定义 如果关系模式R ∈ 2NF，且每个非主属性都不传递依赖于R的每个关系键，则称R属于第三范式（Third Normal Form），简称3NF，记作R ∈ 3NF。

- 第三范式具有如下性质：
  - 1．如果R ∈ 3NF，则R也是2NF。
  - 2．如果R ∈ 2NF，则R不一定是3NF。

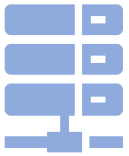
    例如，我们前面由关系模式SCD分解而得到的SD和SC都为2NF，其中，SC∈3NF；但在SD中存在着非主属性MN对主键SNO传递依赖，SD不是3NF。对于SD，应该进一步进行分解，使其转换成3NF



3NF

- 3NF规范化是指把2NF关系模式通过投影分解转换成3NF关系模式的集合。
- 和2NF规范化时遵循的原则相同，即“一事一表”，让一个关系只描述一个实体或者实体间的联系。

- 下面以2NF关系模式SD为例，来说明3NF规范化的过程。



3NF

例： 将SD(SNO,SN,AGE,DEPT,MN)规范到3NF。

- 分析SD的属性组成，可以判断，关系SD实际上描述了两个实体：
  - 一个为学生实体，属性有SNO，SN，AGE，DEPT；
  - 另一个是系的实体，其属性DEPT和MN。
- 根据投影分解的原则，我们可以将SD分解成如下两个关系
  - S(SNO,SN,AGE,DEPT)，描述学生实体；
  - D(DEPT，MN)，描述系的实体。

3NF

S

SNO	SN	AGE	DEPT
S1	赵亦	17	计算机
S2	钱尔	18	信息
S3	孙珊	20	信息
S4	李思	21	自动化

D

DEPT	MN
计算机	刘伟
信息	王平
自动化	刘伟

- 对于分解后的两个关系S和D，主键分别为SNO和DEPT，不存在非主属性对主键的传递函数依赖。因此，S∈3NF，D∈3NF。

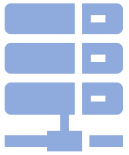
3NF

- 关系模式SD由2NF分解为3NF后，函数依赖关系变得更加简单，既没有非主属性对键的部分依赖，也没有非主属性对键的传递依赖，解决了2NF中存在的四种数据异常问题。
  - 1. 数据冗余降低。系主任的名字存储的次数与该系的学生人数无关，只在关系D中存储一次。
  - 2．不存在插入异常。当一个新系没有学生时，该系的信息可以直接插入到关系D中，而与学生关系S无关。
  - 3．不存在删除异常。要删除某系的全部学生而仍然保留该系的有关信息时，可以只删除学生关系S中的相关学生记录，而不影响系关系D中的数据。

3NF

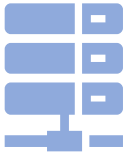
- 关系模式SD由2NF分解为3NF后，函数依赖关系变得更加简单，既没有非主属性对键的部分依赖，也没有非主属性对键的传递依赖，解决了2NF中存在的四种数据异常问题。
  - 4．不存在更新异常。更换系主任时，只需修改关系D中一个相应元组的MN属性值，从而不会出现数据的不一致现象。
- SCD规范到3NF后，所存在的异常现象已经全部消失。

- 那3NF是不是规范化的终点?



3NF

- 3NF只限制了非主属性对键的依赖关系，而没有限制主属性对键的依赖关系。
- 如果发生了这种依赖，仍有可能存在数据冗余、插入异常、删除异常和修改异常。



3NF

- 考虑关系模式SNC（SNO，SN，CNO，SCORE），其中SNO代表学号，SN代表学生姓名并假设没有重名，CNO代表课程号，SCORE代表成绩。可以判定，SNC有两个候选键（SNO，CNO）和（SN，CNO），其函数依赖如下：
  - SNO ↔ SN
  - （SNO，CNO）→SCORE
  - （SN，CNO）→SCORE
- 唯一的非主属性SCORE对键不存在部分函数依赖，也不存在传递函数依赖。所以SNC∈3NF。

3NF

- 因为SNO ↔ SN，即决定因素SNO或SN不包含候选键，从另一个角度说，存在着主属性对键的部分函数依赖：

  (SNO，CNO) $\xrightarrow{P}$ SN，　(SN，CNO) $\xrightarrow{P}$ SNO

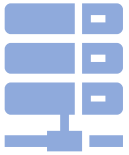
- 由于存在着这种主属性对键的部分函数依赖关系，造成了关系SNC中存在着较大的数据冗余，学生姓名的存储次数等于该生所选的课程数，从而会引起修改异常。
- 若学生没有选课，则无法记录学生的姓名信息，或学号信息，产生插入异常。若学生所有课程都退选了，则会删除学生的学号姓名信息对照信息，产生删除异常。

3NF

- 又例：设关系模式TCS（T，C，S），T表示教师，C表示课程，S表示学生。语义假设是，每一位教师只讲授一门课程；每门课程由多个教师讲授；某一学生选定某门课程，就对应于一确定的教师。根据语义假设，TCS的函数依赖是：

  （S，C）→T，（S，T）→C，T→C。

- 对于TCS，（S，C）和（S，T）都是候选键，两个候选键相交，有公共的属性S。TCS中不存在非主属性，也就不可能存在非主属性对键的部分依赖或传递依赖，所以TCS ∈ 3NF。
- 但对TCS进行分析，依然存在数据异常



3NF

- 数据冗余。虽然每个教师只开一门课，但每个选修该教师该门课程的学生元组都要记录这一信息。
- 插入异常。当某门课程本学期不开，自然就没有学生选修。没有学生选修，因为主属性不能为空，教师上该门课程的信息就无法插入。同样原因，学生刚入校,尚未选课，有关信息也不能输入。
- 删除异常。如果选修某门课程的学生全部毕业，删除学生记录的同时,随之也删除了教师开设该门课程的信息。
- 更新异常。当某个教师开设的某门课程改名后，所有选修该教师该门课程的学生元组都要进行修改，如果漏改某个数据，则破坏了数据的完整性。

3NF

关系TCS实例

T	C	S
T1	C1	S1
T1	C1	S2
T2	C1	S3
T2	C1	S4
T3	C2	S2
T4	C2	S2
T4	C3	S2

分析出现上述数据异常问题的原因在于主属性部分依赖于键，即：（S，T） → C，因此关系模式还继续分解，转换成更高一级的范式，以消除数据库操作中的异常现象。

BCNF

- 如果关系模式R ∈ 1NF，且所有的函数依赖X→Y（Y ∉ X），决定因素X都包含了R的一个候选键，则称R属于BC范式（Boyce-Codd Normal Form），记作R ∈ BCNF。

- BCNF性质
  - 满足BCNF的关系将消除任何属性（主属性或非主属性）对键的部分函数依赖和传递函数依赖。也就是说，如果R ∈ BCNF，则R也是3NF。
  - 如果R ∈ 3NF，则R不一定是BCNF

BCNF

- BCNF规范化
  - BCNF规范化是指把3NF关系模式通过投影分解转换成BCNF关系模式的集合。

- 例　将SNC(SNO,SN,CNO，SCORE)规范到BCNF。
  - 分析SNC数据冗余的原因，是因为在这一个关系中存在两个实体，一个为学生实体，属性有SNO、SN；另一个是选课实体，属性有SNO、CNO和SCORE。

BCNF

- 根据投影分解的原则，我们可以将SNC分解成如下两个关系：
  - S1(SNO,SN)，描述学生实体；
  - S2(SNO,CNO,SCORE)，描述学生与课程的联系。
- 对于S1，有两个候选键SNO和SN，
- 对于S2，主键为（SNO，CNO）。
- 在这两个关系中，无论主属性还是非主属性都不存在对键的部分依赖和传递依赖，S1 ∈ BCNF，S2 ∈ BCNF。

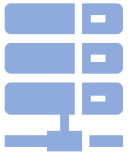
- 观察S1，S2，是否还存在数据异常？

BCNF

- 对TCS模式的规范化：
  - TCS中出现主属性部分依赖于键：（S，T）→ C，T → C
    因此对它进行投影分解，分解为两个关系模式ST（S，T）和TC（T，C），消除函数依赖（S，T）→ C
    - ✓ 数据冗余降低。每个教师开设课程的信息只在TC关系中存储一次。
    - ✓ 不存在插入异常。对于所开课程尚未有学生选修的教师信息可以直接存储在关系TC中，而对于尚未选修课程的学生可以存储在关系ST中。
    - ✓ 不存在删除异常。如果选修某门课程的学生全部毕业，可以只删除关系ST中的相关学生记录，而不影响系关系TC中相应教师开设该门课程的信息。
    - ✓ 不存在更新异常。当某个教师开设的某门课程改名后，只需修改关系TC中的一个相应元组即可，不会破坏数据的完整性。

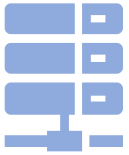
BCNF

- 如果一个关系数据库中所有关系模式都属于3NF，则已在很大程度上消除了插入异常和删除异常，但由于可能存在主属性对候选键的部分依赖和传递依赖，因此关系模式的分离仍不够彻底。
- 如果一个关系数据库中所有关系模式都属于BCNF，那么在函数依赖的范畴内，已经实现了模式的彻底分解，消除了产生插入异常和删除异常的根源，而且数据冗余也减少到极小程度



关系模式的规范化设计

- 规范化理论已经提出了六类范式（有关4NF和5NF的内容不再详细介绍）。
- 各范式级别是在分析函数依赖条件下对关系模式分离程度的一种测度，范式级别可以逐级升高。
- 一个低一级范式的关系模式，通过模式分解转化为若干个高一级范式的关系模式的集合，这种分解过程叫作关系模式的规范化（Normalization）。

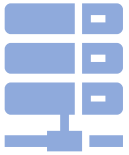


关系模式规范化的目的和原则

- 一个关系只要其分量都是不可分的数据项，就可称作规范化的关系，但这只是最基本的规范化。
- 这样的关系模式是合法的。
- 但人们发现有些关系模式存在插入、删除、修改异常、数据冗余等弊病。
- 规范化的目的就是使结构合理，消除存储异常，使数据冗余尽量小，便于插入、删除和更新。

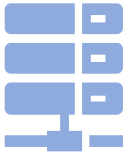
关系模式规范化的步骤

- 规范化就是对原关系进行投影，消除决定属性不是候选键的任何函数依赖。具体可以分为以下几步：
  - 1．对1NF关系进行投影，消除原关系中非主属性对键的部分函数依赖，将1NF关系转换成若干个2NF关系。
  - 2．对2NF关系进行投影，消除原关系中非主属性对键的传递函数依赖，将2NF关系转换成若干个3NF关系。
  - 3．对3NF关系进行投影，消除原关系中主属性对键的部分函数依赖和传递函数依赖，也就是说使决定因素都包含一个候选键。得到一组BCNF关系。



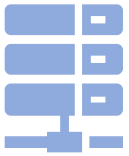
关系模式规范化的步骤

- 一般情况下，我们说没有异常弊病的数据库设计是好的数据库设计，一个不好的关系模式也总是可以通过分解转换成好的关系模式的集合。
- 但是在分解时要全面衡量，综合考虑，视实际情况而定。
  - 对于那些只要求查询而不要求插入、删除等操作的系统，几种异常现象的存在并不影响数据库的操作。这时便不宜过度分解，否则当要对整体查询时，需要更多的多表连接操作，这有可能得不偿失。
  - 允许数据异常存在而换取易用性和性能提升的设计称为非规范化设计
- 在实际应用中，最有价值的是3NF和BCNF，在进行关系模式的设计时，通常分解到3NF就足够了



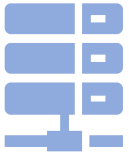
非规范化设计

- 许多数据库应用场景强调性能优先
- 规范化设计有时会导致数据库运行效率的下降
- 非规范化设计
  - 在特殊条件和要求下，适当地降低甚至抛弃关系模式的范式，不再要求一个表只描述一个实体或者实体间的一种联系。其主要目的在于提高数据库的运行效率。



非规范化设计

- 一般认为，在下列情况下可以考虑进行非规范化处理：
  - 大量频繁的查询过程所涉及的表都需要进行连接；
  - 主要的应用程序在执行时要将表连接起来进行查询；
  - 对数据的计算需要临时表或进行复杂的查询。
- 非规范化处理的主要技术包括增加冗余或派生列，对表进行合并、分割或增加重复表等。

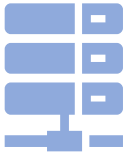


非规范化设计

- 增加冗余列是指在多个表中具有相同的列，它常用来在查询时避免连接操作。例如，如果经常检索一门课的任课教师姓名，则需要做class和teacher表的连接查询：

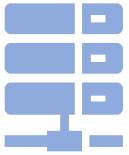
```
select classname,teachername
from class,teacher
where class.teacherNo=teacher.teacherNo
```

- 这样的话就可以在class表中增加一列 teacher_name，就不需要连接操作了。
- 增加冗余列可以在查询时避免连接操作，但它需要更多的存储空间，同时增加表维护的工作量。



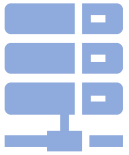
非规范化设计

- 增加派生列指增加的列来自其它表中的数据，由它们计算生成。它的作用是在查询时减少连接操作，避免使用集函数。派生列除于冗余列同样的缺点之外，还需消耗计算资源在数据变化时对派生列的值进行及时维护。



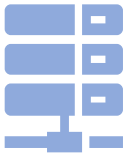
非规范化设计

- 重新组表指如果许多用户需要查看两个表连接出来的结果数据，则把这两个表重新组成一个表来减少连接而提高性能。例如，用户经常需要同时查看课程号，课程名称，任课教师号，任课教师姓名，则可把表 class(classno,classname,teacherno) 和表 teacher(teacherno,teachername) 合并成一个表 class_Teacher(classno,classname,teacherno,teachername)。这样可提高性能，但需要更多的磁盘空间，同时也损失了数据在概念上的独立性



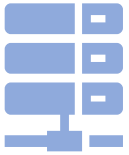
非规范化设计

- 对表做分割可以提高性能，特别是在进行大数据量管理时，常采用“分库分表”策略
- 表分割有两种方式：
  - 水平分割
  - 垂直分割



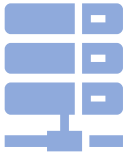
非规范化设计

- 水平分割
  - 根据一列或多列数据的值把数据行放到两个独立的表中。
  - 分割方法包括范围分区、散列分区、列表分区等
  - 水平分割通常在下面的情况下使用：
    - ✓ 表很大，分割后可以降低在查询时需要读的数据和索引的页数，同时也降低了索引的层数，提高查询速度。
    - ✓ 表中的数据本来就有独立性，例如表中分别记录各个地区的数据或不同时期的数据，特别是有些数据常用，而另外一些数据不常用。
  - 需要把数据存放到多个介质上。



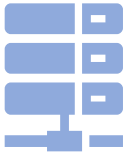
非规范化设计

- 垂直分割
  - 把主键和一些列放到一个表，然后把主键和另外的列放到另一个表中。
  - 如果一个表中某些列常用，而另外一些列不常用，就可以采用垂直分割加快查询速度。其缺点是需要管理冗余列，查询所有数据需要join操作。



非规范化设计

- 非规范化设计的主要优点
  - 减少了查询操作所需的连接
  - 减少了外部键和索引的数量
  - 可以预先进行统计计算，提高了查询时的响应速度
- 非规范化存在的主要问题
  - 增加了数据冗余
  - 影响数据库的完整性
  - 降低了数据更新的速度
  - 增加了存储表所占用的物理空间



非规范化设计

- 无论使用何种非规范化技术，都需要一定的管理来维护数据的完整性
- 管理方式较常采用的是用触发器来实现。对数据的任何修改立即触发对复制列或派生列的相应修改。触发器是实时的，而且相应的处理逻辑只在一个地方出现，易于维护。

作业

- 课本第202页：第1题（多值依赖、4NF不做），第7题第1，2小题
- 下次上课前提交