浙江大学
ZHEJIANG UNIVERSITY

第六章 关系数据库设计理论

陶煜波
计算机科学与技术学院

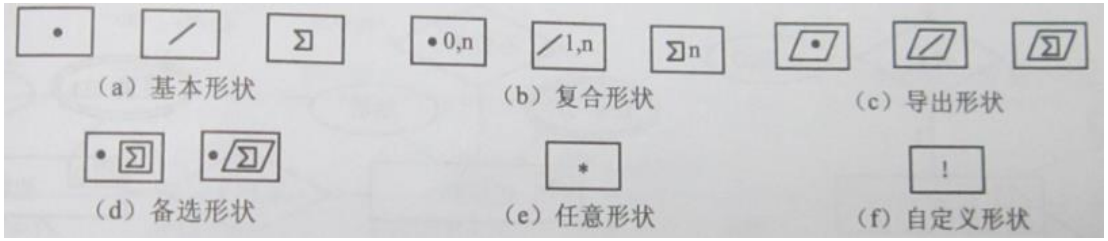
关系数据库设计

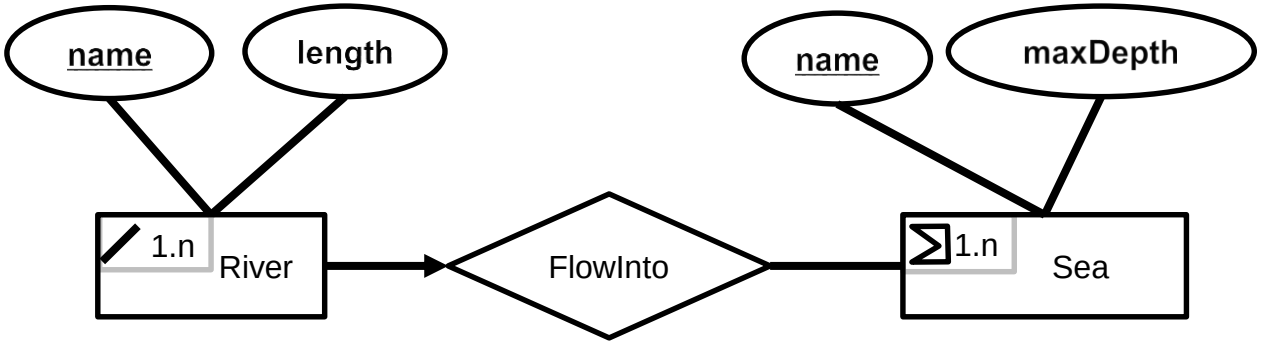
数据库设计一般要经过以下几个步骤：

- 需求分析阶段 → 数据流图和数据字典 (软工)
- 概念结构设计阶段 → E-R图/UML图
- 逻辑结构设计阶段 → 关系数据库模式 (规范化)
- 数据库物理设计阶段 → 存储方式、索引和用户权限
- 数据库实施阶段
- 数据库运行和维护阶段

E/R图

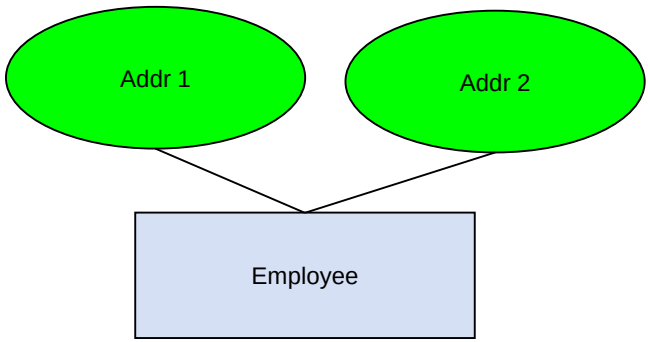
- Primitive units of E/R model
  - Entities sets: classes or types of objects
    - Attributes, Key
    - Pictograms
  - Relationships
    - A subset of cross-product
    - Attributes, Key
    - Multiplicity: 1:1, 1:N, M:N

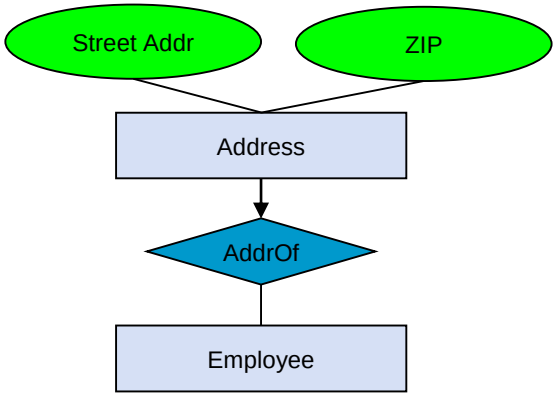




E/R图

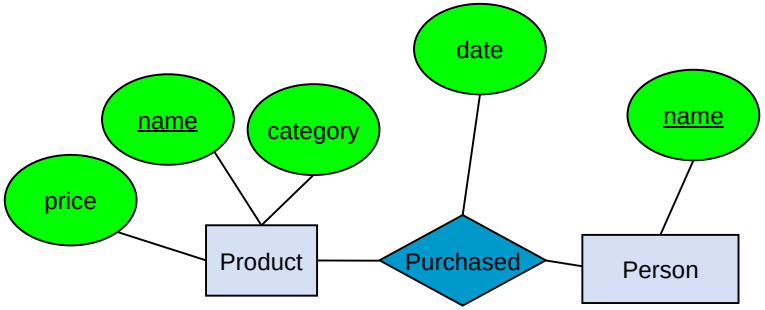
- Design consideration
  - Entity vs. Attributes
    - Multi-valued attributes, such as addresses, phones, and grades

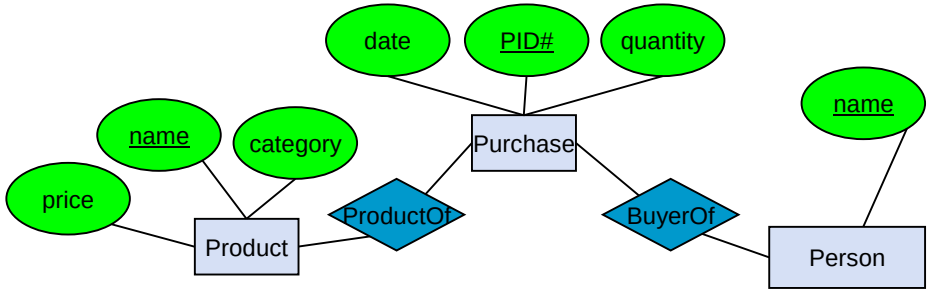




E/R图

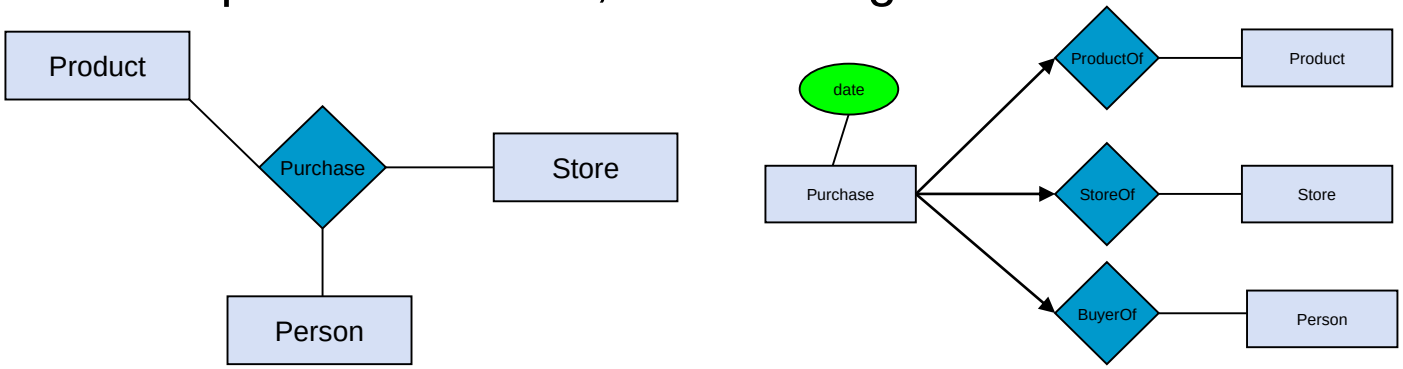
- Design consideration
  - Entity vs. Attributes
    - Multi-valued attributes, such as addresses, phones, and grades
  - Entity vs. Relationship
    - Multiple instances of each entity combination, such as purchase





E/R图

- Design consideration
  - Entity vs. Attributes
    - Multi-valued attributes, such as addresses, phones, and grades
  - Entity vs. Relationship
    - Multiple instances of each entity combination, such as purchase
  - Multi-way vs. New Entity + Binary relationships
    - Multiple combination, more-fine-grained constraints



从E/R图到关系模式

- Translation rules
  - Entity becomes Relation
  - Attributes become columns in the relation
  - Multi-valued attributes become a new relation
    - Includes foreign key to link to relation for the entity
  - Relationships (1:1, 1:N, N:1) become foreign keys
    - Foreign keys enforce unique constraint for 1:1
  - M:N Relationship become a relation
    - Containing foreign keys or relations from participating entities
- Constraints in E/R diagrams
  - Keys: Implicit constraints on uniqueness of entities
  - Single-value constraints
  - Referential integrity constraints: Referenced entities must

从E/R图到关系模式

- 从E-R图到关系模式

方法一： Student (Sno, Sname, ….)

Class(ClassID, num, …)

Class_Student (Sno, ClassID)

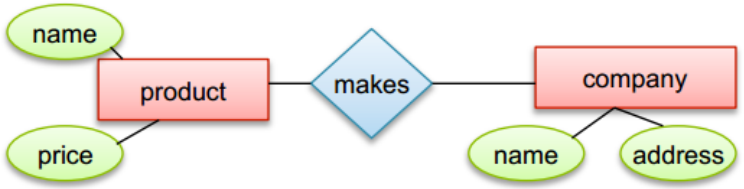
方法二： Student (Sno, Sname, …, ClassID)

Class(ClassID, num, …)

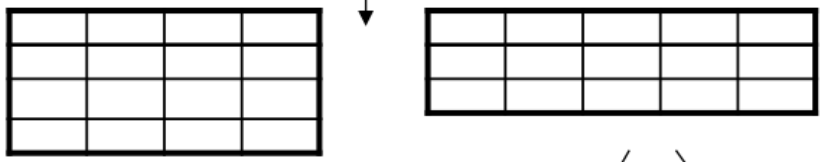
转换得到的关系模式是设计合理或“好”的关系模式？

关系数据库设计

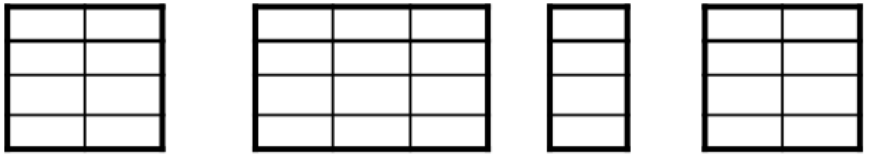
Conceptual Model:



Relational Model:
Tables + constraints
And also functional dep.

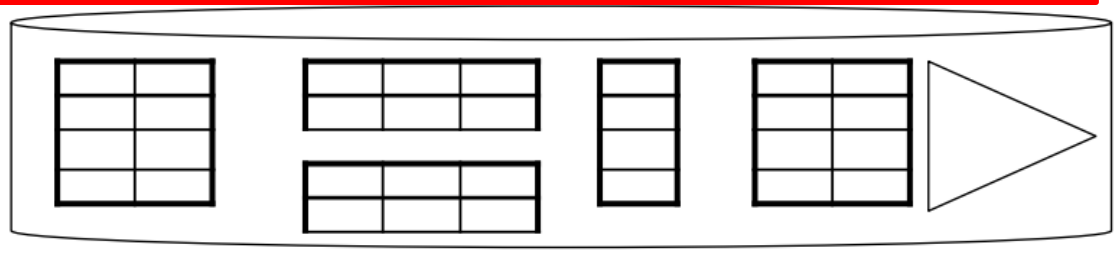


Normalization: 规范化
Eliminates anomalies



Conceptual Schema

Physical storage details



Physical Schema

第六章 关系数据库设计理论

- 6.1 数据依赖对关系模式的影响
- 6.2 函数依赖 (Functional Dependencies)
- 6.3 BC范式 (Boyce-Codd Normal Form)
- 6.4 BC范式和第4范式的局限
- 6.5 小结

参考教材：
数据库系统概念 8.1-8.9

6.1 数据依赖对关系模式的影响

- 数据依赖
  - 通过关系中属性间值的相等与否体现出的数据间的相互关系
  - 现实世界属性间相互联系的抽象
  - 数据的内在性质
  - 语义的体现
- 分类
  - 函数依赖 (Functional Dependencies)
  - 多值依赖 (Multivalued Dependencies)
  - 其他

6.1 数据依赖对关系模式的影响

- 学校数据库(举例)：
  - 学生的学号 (Sno)
  - 所在系 (Sdept)
  - 系主任姓名 (Mname)
  - 课程号 (Cname)
  - 成绩 (Grade)
- E-R图中获得’mega’ relation
  - Student(Sno, Sdept, Mname, Cname, Grade)
  - 属性之间的数据依赖

6.1 数据依赖对关系模式的影响

- 学校数据库的语义：
  - 一个系有若干学生， 一个学生只属于一个系
  - 一个系只有一名主任
  - 一个学生可以选修多门课程， 每门课程有若干学生选修
  - 每个学生所学的每门课程都有一个成绩
- 属性之间的函数依赖
  - Sno → Sdept
  - Sdept → Mname
  - (Sno, Cname) → Grade

6.1 数据依赖对关系模式的影响

- 函数依赖 → 设计异常 (Design anomalies)
  - 数据冗余 (Redundancy)
    - 浪费大量的存储空间
    - 例如，每个系主任的姓名重复出现
  - 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
    - 例如，某系更换系主任后，系统必须修改与该系学生有关的每个元组

6.1 数据依赖对关系模式的影响

- 函数依赖 → 设计异常 (Design anomalies)
  - 插入异常 (Insert anomalies)
    - 有些数据无法正常插入
    - 例如，如果一个系刚成立，尚无学生，就无法把这个系及其系主任的信息存入数据库
  - 删除异常 (Deletion anomalies)
    - 不该删除的数据不得不删
    - 例如，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

6.1 数据依赖对关系模式的影响

- 大学申请数据库(举例)
  - SSN(美国社会安全号)和name
  - Colleges applying to
  - High schools attended (with city)
  - Hobbies
- E-R图中获得'mega' relation
  - Apply(SSN, Sname, Cname, HS, HScity, hobby)
  - 属性之间的数据依赖

Cname，(HS, Hscity), hobby是多值属性 ➔ 多值依赖

6.1 数据依赖对关系模式的影响

- Apply(SSN, Sname, Cname, HS, HScity, hobby)
  - 123 Ann from PAHS (P.A.) and GHS (P.A.) plays tennis and trumpet(喇叭) and applied to Standford, Berkely, and MIT
    - 123, Ann, Stanford, PAHS, P.A., tennis
    - 123, Ann, Berkeley, PAHS, P.A., tennis
    - 123, Ann, Berkeley, PAHS, P.A., trumpet
    - ….

思考：Ann在Apply关系中有多少个元组？

6.1 数据依赖对关系模式的影响

- Apply(SSN, Sname, Cname, HS, HScity, hobby)
- 设计异常
  - 数据冗余
    - HS, HScity
    - Cname, hobby
  - 更新异常
    - Trumpet → Cornet
  - 插入异常
    - 尚未申请学校的学生信息无法插入数据库
  - 删除异常

6.1 数据依赖对关系模式的影响

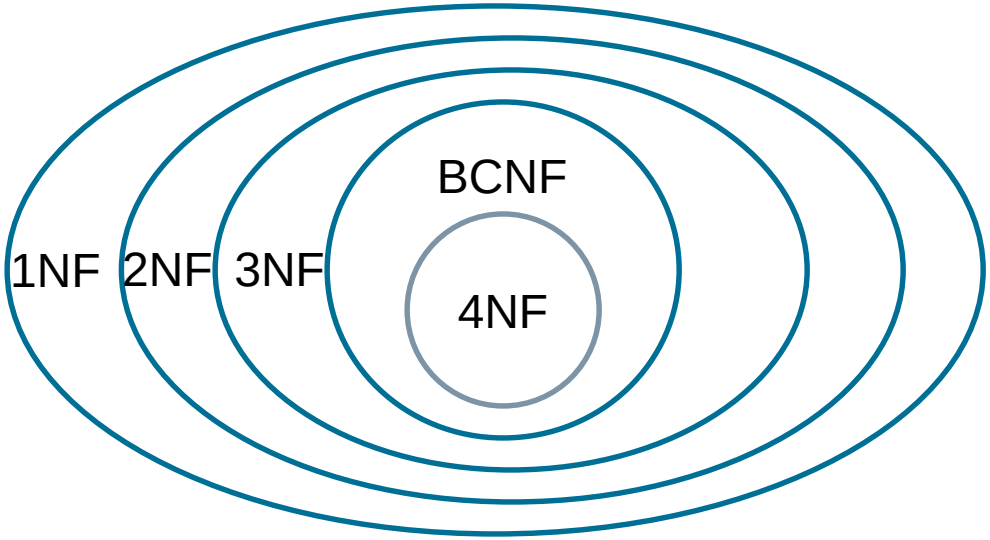
- 结论
  - Student(Sno, Sdept, Mname, Cname, Grade)
  - Apply(SSN, Sname, Cname, HS, HScity, hobby)
  - Student和Apply关系模式不是“好”的模式
  - “好”的模式不会发生插入异常、删除异常、更新异常，数据冗余应尽可能小
- 原因：关系模式中**数据依赖**
- 解决方法：通过**分解**关系模式来消除其中不合适的数据依赖，且能重构原有模式

6.1 数据依赖对关系模式的影响

- Design by decomposition
  - Start with “mega” relations containing everything
  - Decompose into smaller, better relations with the same information
  - Can do decomposition automatically
- Automatic decomposition
  - “Mega” relations + properties of the data
  - System decomposes based on properties
  - Final set of relations satisfies normal form
    - No anomalies, no lost information

6.1 数据依赖对关系模式的影响

- Properties of the data (数据依赖) and Normal Forms (范式)
  - 函数依赖 → BC范式 (Boyce-Codd Normal Forms)
  - 多值依赖 → 第4范式 (Forth Normal Form)



6.1 数据依赖对关系模式的影响

- 1st Normal Form (1NF) = All tables are flat
- *2nd Normal Form = disused*
- **Boyce-Codd Normal Form (BCNF)**
- **3rd Normal Form (3NF)**

DB designs based on ***functional dependencies***, intended to prevent *data anomalies*

- ***4th Normal Forms***

DB designs based on ***multivalued dependencies***

- *5th Normal Forms = see text books*

6.1 数据依赖对关系模式的影响

- 1st Normal Form (1NF)

Student	Majors
Mary	{CS, EE}
Joe	{GIS, SE}
…	…

Violates 1NF

Student	Major
Mary	CS
Mary	EE
Joe	GIS
Joe	SE

In 1st NF

1NF Constraint: Types must be atomic!

6.1 数据依赖对关系模式的影响

- Apply(SSN, Sname, Cname)

  思考：Apply的key？

  - 数据冗余；数据更新和删除异常
  - 每个大学都存储了SSN-Sname信息
- 函数依赖 SSN→ Sname
  - 相同的SSN始终得到相同的Sname
  - f(SSN) = Sname
  - SSN的Sname应该只存储一次
- BC范式： if A→B, then A is a key
- 分解： Student(SSN, Sname), Apply(SSN, Cname)

6.1 数据依赖对关系模式的影响

- Apply(SSN, Cname, HS)　　思考：Apply的key？
  - 数据冗余；数据更新和删除异常
  - Multiplicative effect
    - C个大学，H个高中，C×H个元组，期望是C+H个元组
  - BC范式并没有解决这一问题，无函数依赖
- 多值依赖 SSN →→ Cname (SSN↠HS)
  - Given SSN has every combination of Cname and HS
  - SSN的Cname和HS应该都只存储一次
- 第4范式：if A→→B, then A is a key
- 分解：Apply(SSN, Cname), HighSchool(SSN, HS)

思考：如果C=1和H=1， SSN →→ Cname还成立吗？

6.1 数据依赖对关系模式的影响

- Design by decomposition
  - “Mega” relations + properties of the data
  - System decomposes based on properties
  - Final set of relations satisfies normal form
    - No anomalies, no lost information
  - Functional dependencies ➔ Boyce-Codd Normal Form
  - Multivalued dependences ➔ Fourth Normal Form

第六章 关系数据库设计理论

- 6.1 数据依赖对关系模式的影响
- 6.2 函数依赖 (Functional Dependencies)
- 6.3 BC范式 (Boyce-Codd Normal Form)
- 6.4 BC范式和第4范式的局限
- 6.5 小结

参考教材：
数据库系统概念 8.1-8.9

6.2 函数依赖

- 大学申请数据库(举例)

Student (SSN, Sname, address, HScode, HSname, HScity, GPA, priority)

Apply (SSN, Cname, state, date, major)

- 假设priority取决于GPA
  - GPA> 3.8 ➔ priority = 1
  - 3.3<GPA<=3.8 ➔ priority = 2
  - GPA<=3.3 ➔priority = 3
- 相同GPA的两个元组具有相同的priority

6.2 函数依赖

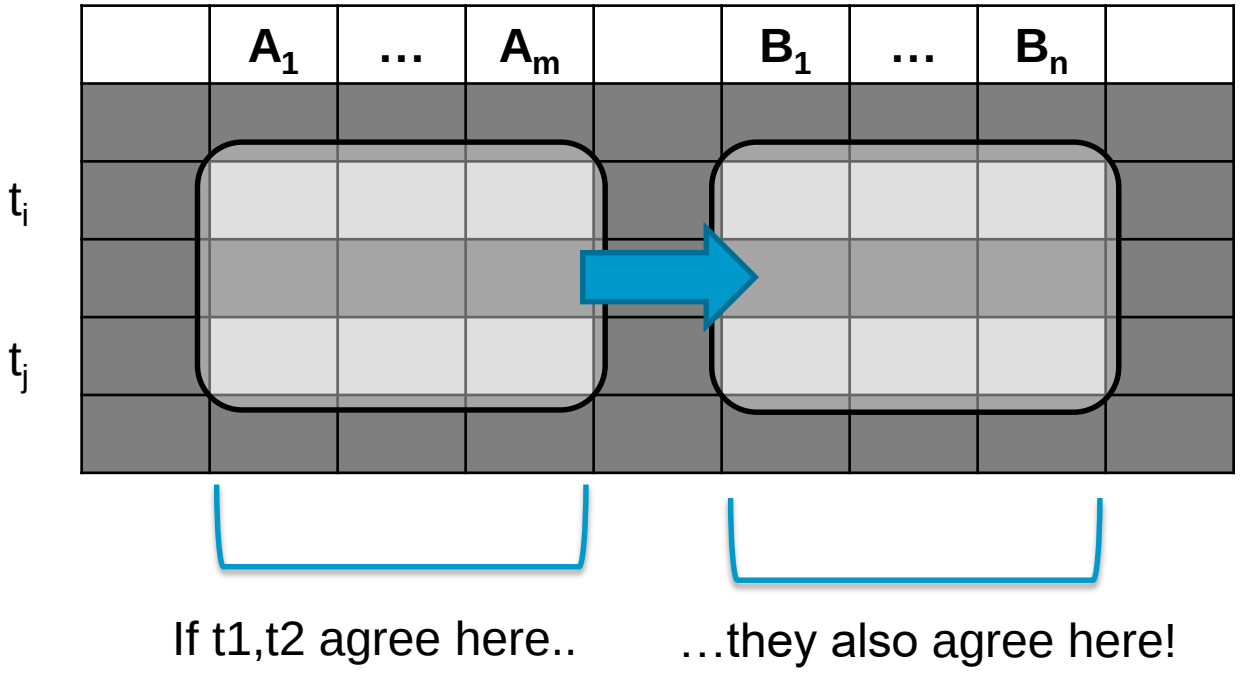
Student (SSN, Sname, address, HScode, HSname, HScity, GPA, priority)

- 相同GPA的两个元组具有相同的priority
- 函数依赖 Functional Dependencies (FDs)
- 关系R中，属性集A和属性集B存在函数依赖的定义(t,u是关系R的元组)：

$$\forall t, u \in R:$$
$$t[A_1, \dots A_n] = u[A_1, \dots A_n] => \ t[B_1, \dots B_m] = u[B_1, \dots B_m]$$

6.2 函数依赖

- 函数依赖从哪儿来？
  - Based on knowledge of real word
    - 需求分析→数据流图&数据字典→E-R图→关系与依赖
  - All instances of relation must adhere
    - R($\bar{A}, \bar{B}, \bar{C}$)，$\bar{A} \rightarrow \bar{B}$



Given attribute sets **A={A_1,…,A_m}** and **B = {B_1,…B_n}** in **R,**

The ***functional dependency*** **A→ B on R** holds if for ***any*** t_i,t_j in R:

if $t_i[A_1] = t_j[A_1]$ AND $t_i[A_2]=t_j[A_2]$ AND … AND $t_i[A_m] = t_j[A_m]$

then $t_i[B_1] = t_j[B_1]$ AND $t_i[B_2]=t_j[B_2]$ AND … AND $t_i[B_n] = t_j[B_n]$

6.2 函数依赖

- 关系R(A, B, C, D, E)具有以下2个函数依赖
  - A,B → C
  - C,D → E

假设A，B，D分别最多有3个不同的取值，E最多有多少个不同的取值？

27

6.2 函数依赖

- 关系Student (SSN, Sname, address, HScode, HSname, HScity, GPA, priority)存在哪些函数依赖？
  - SSN → Sname
  - SSN → address
  - HScode → HSname, HScity
  - HSname, HScity → HScode
  - SSN → GPA
  - GPA → priority
  - SSN → priority

  函数依赖的传递性

6.2 函数依赖

- 关系Apply (SSN, Cname, state, date, major)存在哪些函数依赖？
  - Cname → date
  - SSN, Cname → major ?
  - SSN → state

6.2 函数依赖

- 函数依赖与码(Key)的关系
  - Relations with no duplicates
  - R($\bar{A}$, $\bar{B}$), suppose $\bar{A}$ → all attributes
    - $\bar{A}$是关系R的码

$\bar{A}$	$\bar{B}$
$\bar{a}$	$\bar{b}$
$\bar{a}$	$\bar{b}$

6.2 函数依赖

- 函数依赖分类
  - 平凡函数依赖 (Trivial FD)
    - If $\bar{A} \rightarrow \bar{B}$, $\bar{B} \subseteq \bar{A}$
  - 非平凡函数依赖 (Nontrivial FD)
    - If $\bar{A} \rightarrow \bar{B}$, $\bar{B} \not\subseteq \bar{A}$
  - 完全非平凡函数依赖 (Completely nontrivial FD)
    - If $\bar{A} \rightarrow \bar{B}$, $\bar{B} \cap \bar{A} = \emptyset$

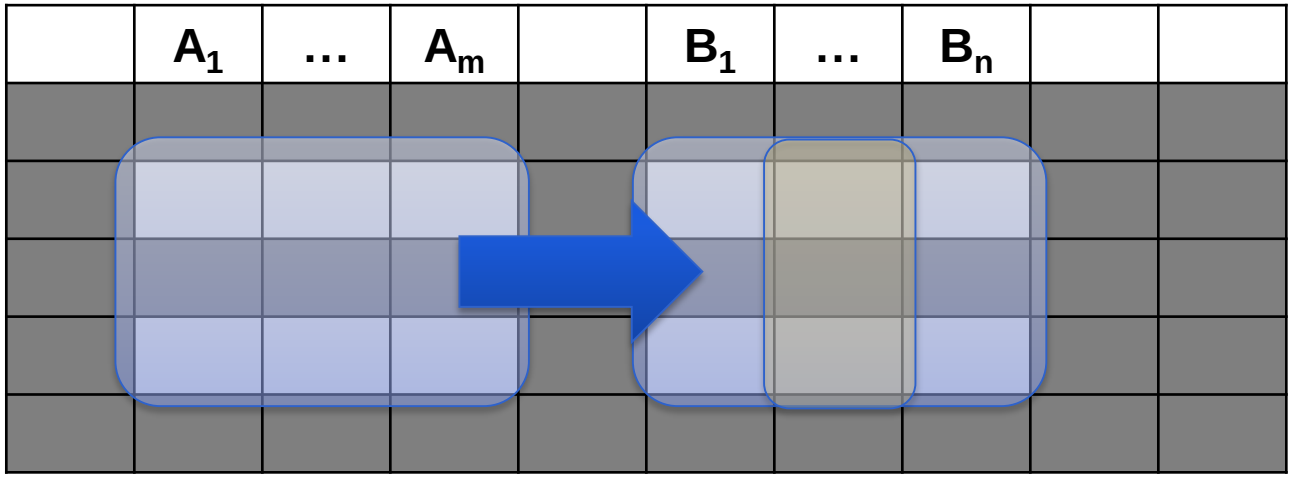
6.2 函数依赖

- High-level idea: why do we care about FDs?
  - Start with some relational schema
  - Find out its functional dependencies (FDs)
  - Use these to design a better schema
    - One which minimizes possibility of anomalies
      - Redundancy
      - Update anomaly
      - Delete anomaly
      - Insert anomaly

6.2 函数依赖

函数依赖规则(Armstrong's Rules)

- 分解规则(Splitting rule)
  - $\bar{A} \rightarrow B_1, \ldots B_m$
  - $\bar{A} \rightarrow B_1, \bar{A} \rightarrow B_2, \ldots, \bar{A} \rightarrow B_m$



6.2 函数依赖

函数依赖规则(Armstrong's Rules)

- 分解规则(Splitting rule)
  - $\bar{A} \rightarrow B_1, \ldots B_m$
  - $\bar{A} \rightarrow B_1, \bar{A} \rightarrow B_2, \ldots, \bar{A} \rightarrow B_m$

~~- $A_1, \ldots A_n \rightarrow \bar{B}$~~
~~- $A_1 \rightarrow \bar{B}, A_2 \rightarrow \bar{B}, \ldots, A_n \rightarrow \bar{B}$~~

  - HSname, HScity → HScode

6.2 函数依赖

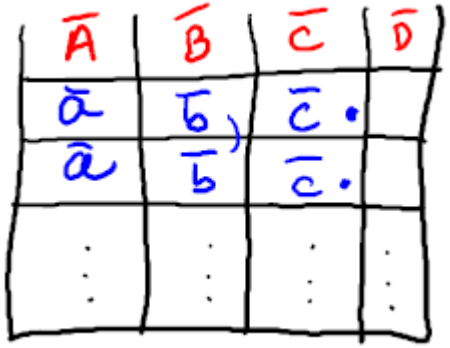
函数依赖规则

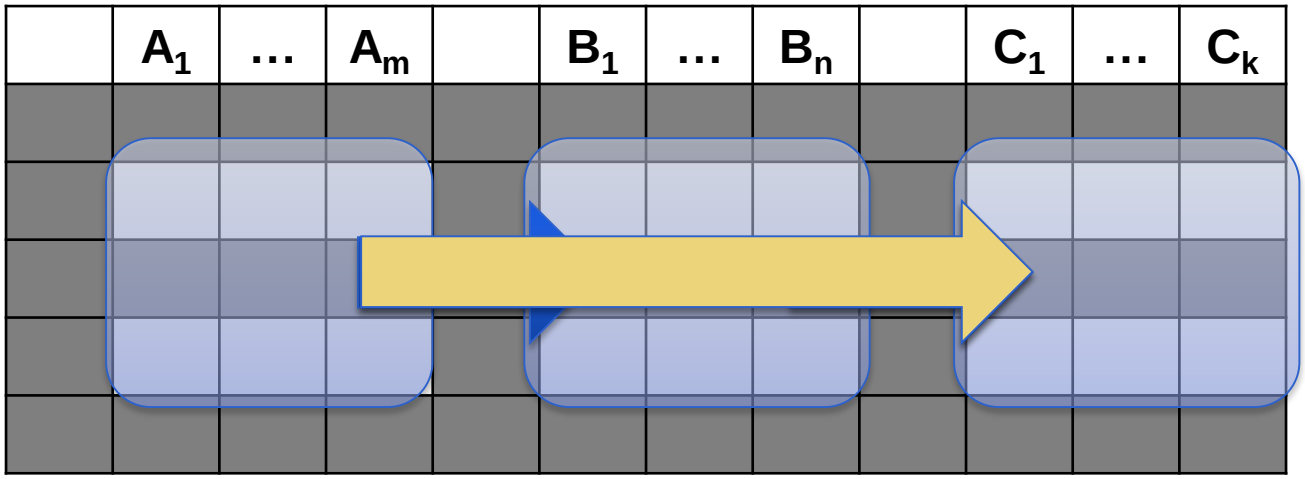
- 分解规则(Splitting rule)
- 合并规则 (Combining rule)
  - $\bar{A} \rightarrow B_1, \bar{A} \rightarrow B_2, \ldots, \bar{A} \rightarrow B_m$
  - $\bar{A} \rightarrow B_1, \ldots B_m$
- 平凡依赖规则 (Trivial-dependency rules)
  - If $\bar{A} \rightarrow \bar{B}$, $\bar{B} \subseteq \bar{A}$
  - If $\bar{A} \rightarrow \bar{B}$, then $\bar{A} \rightarrow \bar{A} \cup \bar{B}$
  - If $\bar{A} \rightarrow \bar{B}$, then $\bar{A} \rightarrow \bar{A} \cap \bar{B}$

6.2 函数依赖

函数依赖规则

- 分解规则(Splitting rule)
- 合并规则 (Combining rule)
- 平凡依赖规则 (Trivial-dependency rules)
- 传递规则 (Transitive rule)
  - If $\bar{A} \to \bar{B}$, $\bar{B} \to \bar{C}$, then $\bar{A} \to \bar{C}$





6.2 函数依赖

- 属性的闭包 (Closure of Attributes)
  - Given relation, FDs, set of attributes $\bar{A}$
  - Find all B such that $\bar{A} \rightarrow B$
  - $\bar{A}^+$ $\{A_1, \ldots, A_n\}^+$

  - 闭包计算算法

    Start with $\{A_1, \ldots, A_n\}$

    Repeat until no change:

    if $\bar{A} \rightarrow \bar{B}$, and $\bar{A}$ in set

    add $\bar{B}$ to set

6.2 函数依赖

- Student (SSN, Sname, address, HScode, HSname, HScity, GPA, priority)
  - SSN → Sname, address, GPA
  - GPA → priority
  - HScode → HSname, HScity
- 属性闭包计算 $\{SSN, HScode\}^+ =$

{SSN, HScode, Sname, address, GPA, priority, HSname, HScity}

思考： {SSN, HScode}是关系Student的码？

6.2 函数依赖

闭包和码

- Is $\bar{A}$ a key for R? (Given R and FDs)
  - Compute $\bar{A}^+$ if $\bar{A}^+$ = all attributes, then $\bar{A}$ is a key
- How can we find all keys given a set of FDs
  - Consider every subset of attributes as $\bar{A}$
  - If $\bar{A}^+$ = all attributes, then $\bar{A}$ is a key
  - Minimum set → increasing size of the subset

6.2 函数依赖

如何为关系指定函数依赖？

- S1 and S2 sets of FDs
- S2 “follows from” S1 if every relation instance satisfying S1 also satisfies S2
  - S1 = {SSN → GPA, GPA → priority}
  - S2 = {SSN → priority}
- How to test? Does $\bar{A} \rightarrow \bar{B}$ follow from S?
  - Compute $\bar{A}^{+}$ base S check if $\bar{B}$ in set
  - Armstrong’s Axioms

思考：数据库设计时，只需要保留函数依赖S1或S2，还是S1和S2?

6.2 函数依赖

- 如何为关系指定函数依赖？
  - Minimal set of completely nontrivial FDs such that all FDs that hold on the relation follow from the dependencies in the set

6.2 函数依赖

- 需求分析→…→关系和函数依赖
- 函数依赖定义
  - 来源于真实世界(应用需求), 所有元组都要满足
- 函数依赖、闭包与码的关系
  - 计算属性的闭包判断是否为码(Key)
- 函数依赖分类（三类）
- 函数依赖规则
  - splitting, combining, trivial-dependency, transitive
- 如何指定函数依赖(E-R图→关系, 那么FDs？)

第六章 关系数据库设计理论

- 6.1 数据依赖对关系模式的影响
- 6.2 函数依赖 (Functional Dependencies)
- 6.3 BC范式 (Boyce-Codd Normal Form)
- 6.4 BC范式和第4范式的局限
- 6.5 小结

参考教材：
数据库系统概念 8.1-8.9

6.3 BC范式

- 关系模式的分解

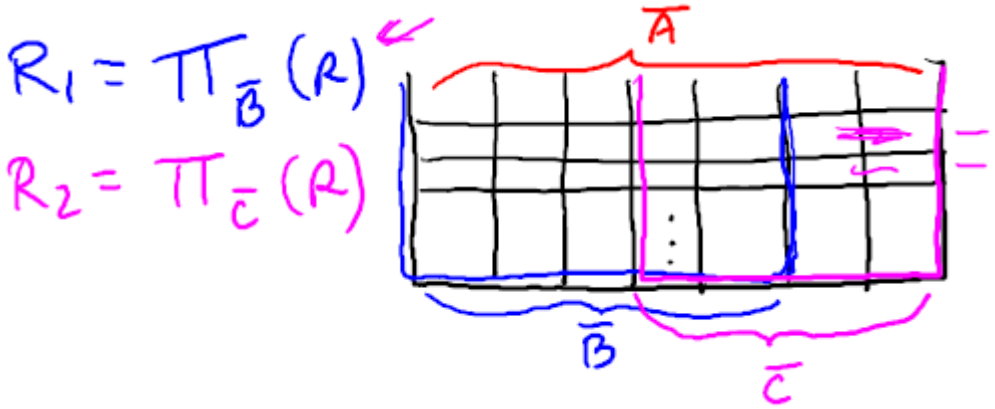
R(A_1, …, A_n) $\bar{A}$

R1(B_1, …, B_k) $\bar{B}$

R2(C_1, …, C_m) $\bar{C}$

$\bar{B} \cup \bar{C} = \bar{A}$

R1 ⋈ R2 = R



6.3 BC范式

$\bar{B} \cup \bar{C} = \bar{A}$

R1 ⋈ R2 = R

例1

Student (SSN, Sname, address, HScode, HSname, HScity, GPA, priority)

以下分解正确(即没有信息丢失)?

S1 (SSN, Sname, address, HScode, GPA, priority)

S2 (HScode, HSname, HScity)

YES!

6.3 BC范式

$\bar{B} \cup \bar{C} = \bar{A}$

R1 ⋈ R2 = R

例2

Student (SSN, Sname, address, HScode, HSname, HScity, GPA, priority)

以下分解正确(即没有信息丢失)?

S1 (SSN, Sname, address, HScode, HSname, HScity)

S2 (Sname, HSname, GPA, priority)

NO!

6.3 BC范式

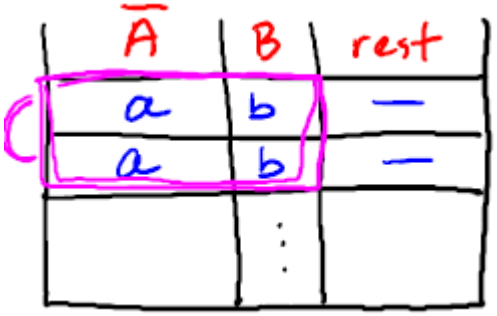
- 基于分解的关系模式设计
  - “Mega” relations + properties of the data
  - System decomposes based on properties
    - “Good” decompositions only (Lossless join property)
    - Into “good” relations (即BCNF)

- A decomposition R to (R1, R2) is lossless if R = R1 Join R2

6.3 BC范式

- Boyce-Codd Normal Form 定义
  - Relation R with FDs is in BCNF if:

    For each $\bar{A} \rightarrow \bar{B}$, $\bar{A}$ is a key
  - BCNF violation



  - $\bar{A}$ contains a key, $\bar{A}$ is a superkey

6.3 BC范式

- 关系Student是否属于BCNF？

Student (SSN, Sname, address, HScode, HSname, HScity, GPA, priority)

  - SSN → Sname, address, GPA
  - GPA → priority　　NO!
  - HScode → HSname, HScity

- 如何判断？
  - 每个函数依赖的左端是否都包含码？
  - Keys: {SSN, HScode}

6.3 BC范式

- 关系Apply是否属于BCNF？

Apply (SSN, Cname, state, date, major)

- SSN, Cname, state → date, major

YES!

- Cname → state
- SSN, Cname → date, major

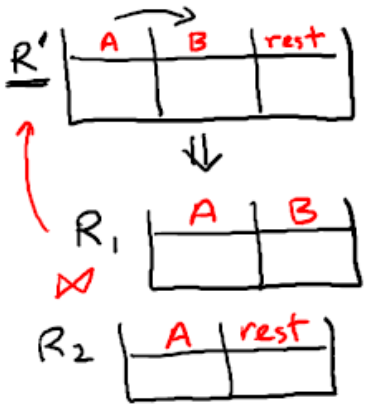
NO!

6.3 BC范式

BCNF decomposition algorithm

Input: relation R + FDs for R

Output: decomposition of R into BCNF relations with "lossless join"



Compute keys for R (using FDs)

Repeat until all relations are in BCNF:

- Pick any R' with $\bar{A} \rightarrow \bar{B}$ that violates BCNF
- Decompose R' into $R_1(\bar{A}, \bar{B})$ and $R_2(\bar{A}, rest)$
- Compute FDs for R_1 and R_2
- Compute keys for R_1 and R_2

6.3 BC范式

例1

Student (SSN, Sname, address, HScode, HSname, HScity, GPA, priority)

- SSN → Sname, address, GPA
- GPA → priority
- HScode → HSname, HScity

S1 (HScode, HSname, HScity)

S2 (GPA, priority)

S3 (SSN, Sname, address, GPA)

S4 (SSN, HScode)

6.3 BC范式

- Student (SSN, Sname, address, HScode, HSname, HScity, GPA, priority)
  - FD1: SSN → Sname, address, GPA
  - FD2: GPA → priority
  - FD3: HScode → HSname, HScity
- Key: {SSN, HScode}，违背BCNF：FD1, FD2, FD3
  - 选择FD3
    - R1 (HScode, HSname, HScity)
    - R2 (HScode, SSN, Sname, address, GPA, priority)
      - Key: {SSN, HScode}
      - FDs: FD1, FD2 (都违背BCNF)
      - 选择FD2
        - » R3 (GPA, priority)
        - » R4 (HScode, GPA, SSN, Sname, address)

6.3 BC范式

- Student (SSN, Sname, address, HScode, HSname, HScity, GPA, priority)
  - FD1: SSN → Sname, address, GPA
  - FD2: GPA → priority
  - FD3: HScode → HSname, HScity
- R1 (HScode, HSname, HScity)
- R3 (GPA, priority)
- R4 (HScode, GPA, SSN, Sname, address)
  - Key: {HScode, SSN}
  - FDs: FD1 (违背BCNF)
    - 选择FD1分解
      - R5 (SSN, Sname, address, GPA)
      - R6 (SSN, HScode)

6.3 BC范式

- Student (SSN, Sname, address, HScode, HSname, HScity, GPA, priority)
  - FD1: SSN → Sname, address, GPA
  - FD2: GPA → priority
  - FD3: HScode → HSname, HScity
- Key: {SSN, HScode}，违背BCNF：FD1, FD2, FD3
  - 选择FD1
    - R1 (SSN, Sname, address, GPA)
    - R2 (SSN, HScode, HSname, HScity, priority)
      - Key: {SSN, HScode, priority}
      - FDs: FD3 (违背BCNF)
      - 选择FD3
        - » R3 (HScode, HSname, HScity)
        - » R4 (HScode, SSN, priority)

6.3 BC范式

- Student (SSN, Sname, address, HScode, HSname, HScity, GPA, priority)
  - FD1: SSN → Sname, address, GPA
  - FD2: GPA → priority
  - FD3: HScode → HSname, HScity
- Key: {SSN, HScode}，违背BCNF：FD1, FD2, FD3
  - 选择FD1
    - R1 (SSN, Sname, address, GPA)
    - R2 (SSN, HScode, HSname, HScity, priority)
      - Key: {SSN, HScode~~, priority~~}
      - FDs: FD3, FD4: SSN → priority (都违背BCNF)
      - 选择FD3
        - » R3 (HScode, HSname, HScity)
        - » R4 (HScode, SSN, priority)

6.3 BC范式

- Student (SSN, Sname, address, HScode, HSname, HScity, GPA, priority)
  - FD1: SSN → Sname, address, GPA
  - FD2: GPA → priority
  - FD3: HScode → HSname, HScity
- R1 (SSN, Sname, address, GPA)
- R3 (HScode, HSname, HScity)
- R4 (HScode, SSN, priority)
  - Key: {HScode, SSN}
  - FDs: FD4: SSN → priority (违背BCNF)
  - 选择FD4分解
    - R5 (SSN, priority) → 丢失了FD2
    - R6 (SSN, HScode)

6.3 BC范式

- Student (SSN, Sname, address, HScode, HSname, HScity, GPA, priority)
  - FD1: SSN → Sname, address, GPA
  - FD2: GPA → priority
  - FD3: HScode → HSname, HScity
- Key: {SSN, HScode}，违背BCNF：FD1, FD2, FD3
  - 选择FD1 (A→B, A→BA$^+$)
    - R1 (SSN, Sname, address, GPA, priority)
      - Key: {SSN}
      - FDs: FD2 (违背BCNF)
        - » R3 (GPA, priority)
        - » R4 (SSN, Sname, address, GPA)
    - R2 (SSN, HScode, HSname, HScity)

6.3 BC范式

- Student (SSN, Sname, address, HScode, HSname, HScity, GPA, priority)
  - FD1: SSN → Sname, address, GPA
  - FD2: GPA → priority
  - FD3: HScode → HSname, HScity
- Key: {SSN, HScode}，违背BCNF：FD1, FD2, FD3
  - 选择FD1 (A→B, A→BA$^+$)
    - R1 (SSN, Sname, address, GPA, priority)
      - R3 (GPA, priority)
      - R4 (SSN, Sname, address, GPA)
    - R2 (SSN, HScode, HSname, HScity)
      - Key: {SSN, HScode}
      - FDs: FD3 (违背BCNF)
        - » R5 (HScode, HSname, HScity), R6(SSN, HScode)

6.3 BC范式

例2

Apply (SSN, Cname, state, date, major)　　B

- Cname → state
- SSN, Cname → date, major

A. Apply(SSN,cName,state,date,major)

B. A1(cName,state), A2(SSN,cName,date,major)

C. A1(cName,state), A2(SSN,date,major)

D. A1(cName,state), A2(SSN,cName,date), A3(SSN,cName,major)

6.3 BC范式

Compute keys for R (using FDs)

Repeat until all relations are in BCNF:

Pick any R' with $\bar{A} \rightarrow \bar{B}$ that violates BCNF

Decompose R' into $R_1(\bar{A}, \bar{B})$ and $R_2(\bar{A}, rest)$

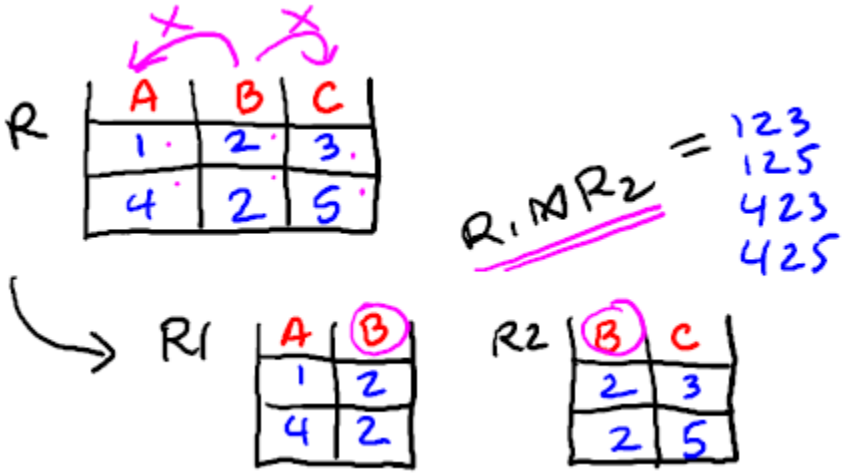
Compute FDs for R_1 and R_2

Compute keys for R_1 and R_2

- 选择不同的R'，最终分解结果是否相同？
- Extend: A → B，A → BA^+
- Implied FDs closure

6.3 BC范式

- Does BCNF guarantee a good decomposition?
  - Removes anomalies? YES!
  - Can logically reconstruct original relation?
    - Too few or too many tuples?



第六章 关系数据库设计理论

- 6.1 数据依赖对关系模式的影响
- 6.2 函数依赖 (Functional Dependencies)
- 6.3 BC范式 (Boyce-Codd Normal Form)
- 6.4 BC范式和第4范式的局限
- 6.5 小结

参考教材：
数据库系统概念 8.1-8.9

6.4 BC范式和第4范式的局限

- BC范式
  - Relation R with FDs is in BCNF if:

    For each A→B, A is a key
- 4范式
  - Relation R with MVDs is in 4NF if:

    For each nontrivial A→→B, A is a key

6.4 BC范式和第4范式的局限

例1

- Apply (SSN, Cname, date, major)
  - Can apply to each college once for one major
  - Colleges have non-overlapping application dates

函数依赖在分解后的关系中丢失

- FDs: SSN, Cname → date, major date → Cname
- Keys: SSN, Cname
- BCNF: NO. R1(date, Cname), R2(SSN, date, major)
- Good design? Not necessarily. 3rd Normal Form

6.4 BC范式和第4范式的局限

例2

- Student (SSN, HSname, GPA, priority)
  - Multiple HS okay, priority determined from GPA

函数依赖在分解后的关系中丢失

- FDs: SSN → GPA GPA → priority SSN → priority
- Keys: SSN, HSname

选择不同函数依赖顺序会得到不同分解结果

- BCNF: NO. R1(SSN, priority), R2(SSN, GPA), R3(SSN, HSname)
- Good design? Not necessarily. 3rd Normal Form

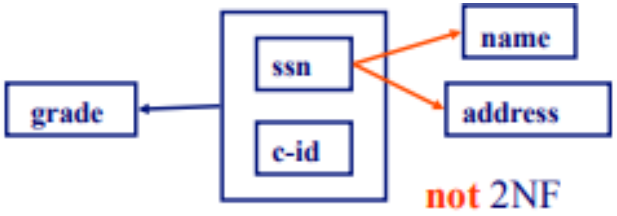
6.4 BC范式和第4范式的局限

- 1st Normal Form (1NF)
  - Attributes are atomic (no set-valued attr.)

Ssn	Name	Dependents
123	Smith	Peter Mary John
234	Jones	Ann Michael

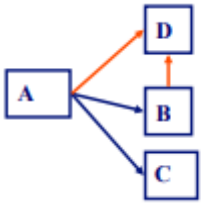
not 1NF

- 2nd Normal Form (2NF)
  - 1NF + non-key attributes fully depend on the key
  - Example: Takes(ssn, cid, grade, name, address)
    - ssn → name, address
    - ssn, cid → grade



not 2NF

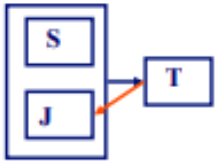
- 3rd Normal Form (3NF)
  - 2NF + no transitive depdencies



in 2NF, but not in 3NF

6.4 BC范式和第4范式的局限

- A relation R is in 3NF if for every FD A → B
  - It is trivial (If $\bar{A} \rightarrow \bar{B}$, $\bar{B} \subseteq \bar{A}$)
  - A is a superkey (good FDs)
  - B is part of a candiate key (A may not be a superkey)
- STJ(Student, Teacher, subJect)



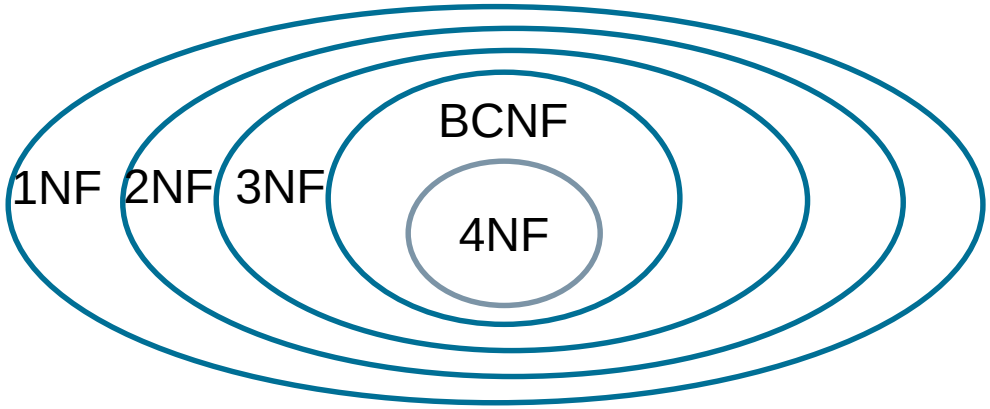
  - S, J → T, T → J
  - BCNF: R1(T, J), R2(S, T) [dependency perserving?]
  - 3NF forgives the red arrow
- In practice, aim for
  - BCNF: lossless join, dependency preservation
  - If impossible, 3NF: lossless join, dependency preservation

6.4 BC范式和第4范式的局限

- BC范式
  - Relation R with FDs is in BCNF if:

    For each A→B, A is a key
- 4范式
  - Relation R with MVDs is in 4NF if:

    For each nontrivial A→→B, A is a key
- After decomposition, no guarantee dependencies can be checked on decomposed relations



6.4 BC范式和第4范式的局限

例3

"Denormalized" relation

- Scores (SSN, Sname, SAT, ACT)
  - Multiple SATs and ACTs allowed
  - All queries return name + composite score for SSN (此时更倾向于原始关系，包含所有属性)

- FDs + keys: SSN → Sname. Key(SSN, SAT, ACT)
- MVDs: SSN, Sname →→ SAT
- 4NF: NO. R1(SSN, Sname), R2(SSN, SAT), R3(SSN, ACT)

6.4 BC范式和第4范式的局限

例4

- College (Cname, state)
- CollegeSize (Cname, enrollment)
- CollegeScores (Cname, avgSAT)
- CollegeGrades (Cname, avgGPA)
- …

Too decomposed

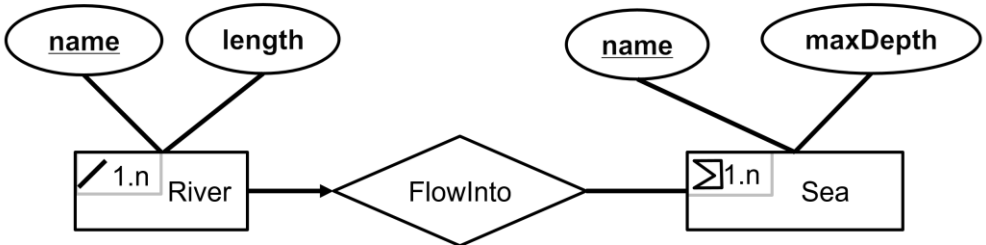
- BCNF/4NF? Yes.
- Good design? Not necessarily.
  - 可用其他4范式关系表达，且包含信息相同

6.4 BC范式和第4范式的局限

- E/R图转换关系模式时，1:1 / 1:N联系尽量和弱实体 / N端实体关系合并，减少关系数量，同时保证所有关系都属于BCNF
  - River(name, length, shape(MultiLineString))
    - name → length, shape

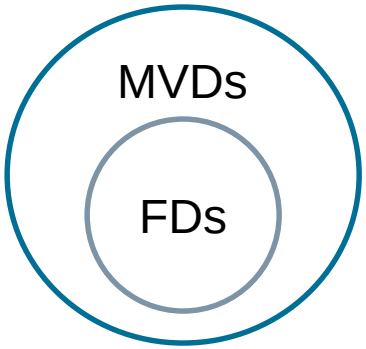
思考：几何属性适合作为主码(primary key)吗？

  - FlowInto(riverName, SeaName)
    - riverName → SeaName
  - River(name, length, shape(MultiLineString), seaName reference Sea)
  - Sea(name, maxDepth, shape(MultiPolygon)



6.5 关系数据库设计理论小结

- Designing a database schema
  - Usually many designs possible
  - Some are (much) better than others!
  - How do we choose?



- Very nice theory for relational DB design
  - Normal forms – “good” relations
  - Design by decomposition
  - Usually intuitive and works well
  - Some shortcomings (Dependency enforcement, Query workload, Over-decomposition)