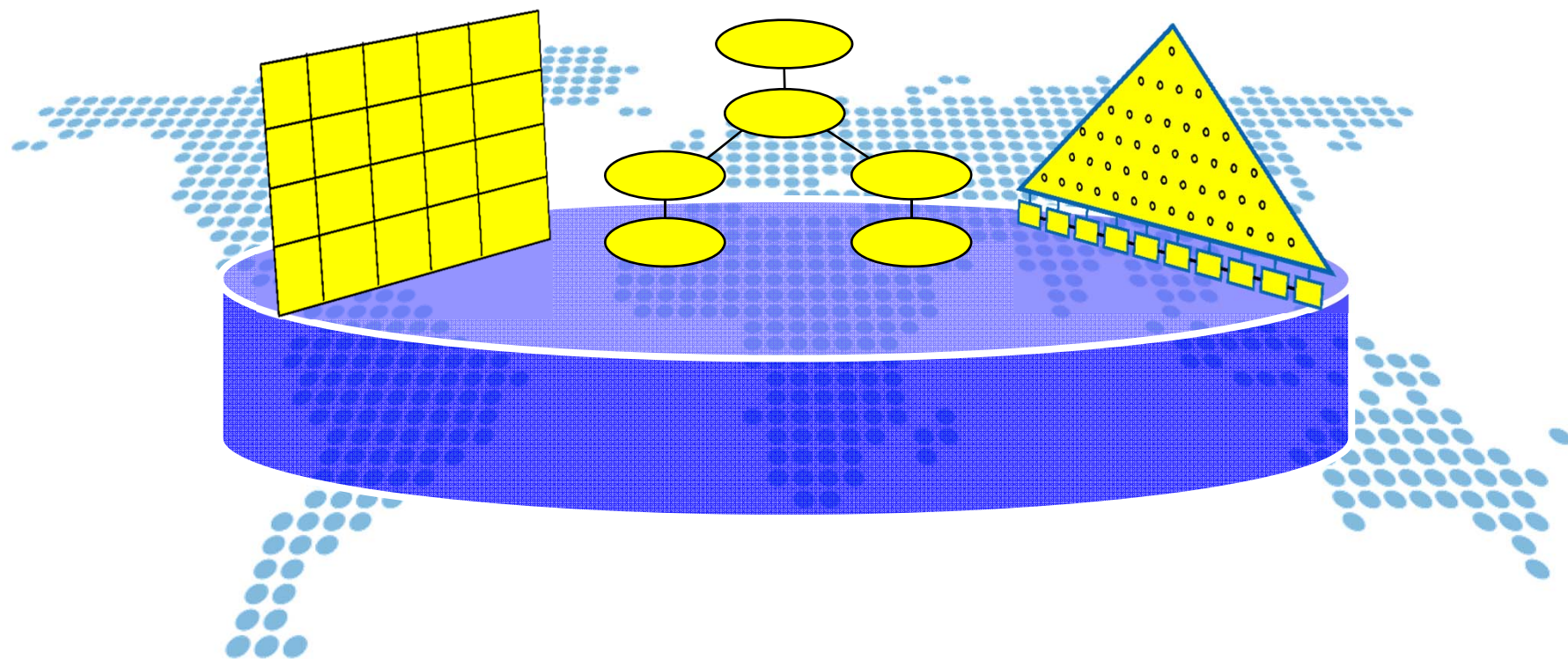


数据库系统

# 关系模型与SQL (4)

陈世敏

(中科院计算所)



# 前面内容

- ER模型，关系模型，ER→关系模型
- SQL 初步
  - 记录的增删改
- 简单的查询+关系代数
  - 集合操作：并、交、差
  - 选择行或列：选择、投影
  - 两个关系元组之间的操作：笛卡尔积、连接
  - 其它：重命名、除
- 丰富的SQL Select功能+扩展关系代数
  - 扩展关系代数
  - 单个Select语句：aggregation, group by, having, order by
  - 嵌套Select语句：in, exists, unique, op ANY/ALL
- 完整性约束
  - domain, unique, primary key, not null
  - foreign key, 执行
  - check, Assertion, trigger

# 数据库设计过程

- 1) 需求分析
- 2) 概念设计 (ER模型)
- 3) 逻辑设计 (ER模型 → 关系模型)
- 4) 模式细化 (规范化等)
- 5) 物理设计 (物理模式, 性能等)
- 6) 应用与安全设计 (定义外部模式等)

# Outline

- 模式细化：范式
  - 数据冗余的问题
  - 范式介绍
  - 函数依赖与2NF,3NF,BCNF
  - 分解为BCNF
  - 多值依赖和连接依赖
- 物理设计：索引的增删
- 外部模式设计：视图

# 数据冗余的问题

- 冗余：同样的数据在数据库中被存储了多次
- 冗余带来的问题
  - 冗余存储
  - 更新异常
  - 插入异常
  - 删除异常

# 数据冗余的问题：举例

Employee

ID	Name	Rating	Wage
1	Alan	5	30
2	Bob	7	50
3	Carol	7	50
4	Dan	5	30

- 假设隐含的条件：相同的rating，工资wage也相同
- 问题
  - 冗余存储：例如rating=5与wage=30之间的关系存了2次
  - 实际上耗费了更多的存储空间

# 数据冗余的问题：举例

Employee

ID	Name	Rating	Wage
1	Alan	5	30→40
2	Bob	7	50
3	Carol	7	50
4	Dan	5	30

- 假设隐含的条件：相同的rating，工资wage也相同
- 问题
  - 更新异常：修改wage时，必须修改所有rating相同的记录，否则就不一致了
  - 例如：下面的update语句破坏了隐含条件，导致了数据异常  
update Employee  
set Wage= 40  
where ID=1;

# 数据冗余的问题：举例

Employee

ID	Name	Rating	Wage
1	Alan	5	30
2	Bob	7	50
3	Carol	7	50
4	Dan	5	30
5	Eva	5	40

- 假设隐含的条件：相同的rating，工资wage也相同
- 问题
  - 插入异常：如果不知道rating对应的wage，就无法正确插入
  - 例如：下面的insert语句破坏了隐含条件，导致了数据异常  
insert into Employee values (5, 'Eva', 5, 40);



# 数据冗余的问题：举例

Employee

ID	Name	Rating	Wage
1	Alan	5	30
2	Bob	7	50
3	Carol	7	50
4	Dan	5	30

- 假设隐含的条件：相同的rating，工资wage也相同

- 问题

- ❑ 删除异常：删除了所有具有rating=5的值时，  
关于rating=5的wage信息也消失了！

- ❑ 例如：

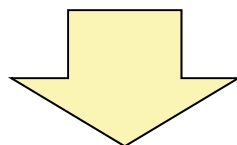
- delete from Employee  
where Rating=5;

# 解决方法：模式分解（我们将学习）

**Employee**

ID	Name	Rating	Wage
1	Alan	5	30
2	Bob	7	50
3	Carol	7	50
4	Dan	5	30

可以通过分解模式来消除冗余



**Employee**

ID	Name	Rating
1	Alan	5
2	Bob	7
3	Carol	7
4	Dan	5

**WageRate**

Rating	Wage
5	30
7	50

# 下面的内容

- 冗余类型和范式

- 每种范式是什么？
- 怎么判断一个关系模式是否满足某种范式？
- 如何通过模式分解使一个关系表满足希望的范式？

# 冗余的类型

- 函数依赖 (Functional Dependency, FD)
- 多值依赖 (Multivalued Dependency, MVD)
  - FD是一种特殊的MVD
- 连接依赖 (Join Dependency, JD)
  - MVD是一种特殊的JD
- 我们会一一介绍

# 范式 (Normal Form)

- 范式就是规范的形式
- 关系模式的每种范式：消除了一种冗余
- 模式求精细化需要使设计符合一定的范式

# 范式（Normal Form）

**1NF:** 所有属性（列）都是原子类型

**2NF:** 消除函数依赖中的部分依赖

**3NF:** 消除函数依赖中的非键传递依赖

**BCNF:** 消除所有函数依赖（希望达到）

**4NF:** 消除多值依赖

**5NF:** 消除连接依赖

**6NF**等其它范式(不介绍)

# 讲解将用传统关系代数

- 在关系模式设计时，消除冗余
- 那么，关系将是一个集合，而不是包/多集
- 所以，我们下面都将用传统关系代数
  - 主要考虑基础的关系表的设计
  - 通常基础表定义了主键，所以没有多个相同记录
  - 不关注运算过程中的中间结果
  - 所以实际上不需要扩展关系代数

# Outline

- 逻辑设计：ER图到关系模型
- 模式细化：范式
  - 数据冗余的问题
  - 范式介绍
  - 函数依赖与2NF,3NF,BCNF
    - 什么是函数依赖？
    - 怎样简化函数依赖？
    - 2NF, 3NF, BCNF
  - 分解为BCNF
  - 多值依赖和连接依赖
- 物理设计和外部模式设计：视图，索引



# 函数依赖 (FD)

- 前提条件

- $R(U)$  是属性集  $U = \{A_1, \dots, A_m\}$  上的一个关系模式
- $X \subseteq U, Y \subseteq U$

- 函数依赖  $X \rightarrow Y$

- 任意两个元组在  $X$  上取值相同  $\Rightarrow$  它们在  $Y$  上取值也相同
- 形式化表述：对于任意的实例  $r$ ,  $\forall t_1, t_2 \in r$ , 若  $t_1[X] = t_2[X]$ , 则  $t_1[Y] = t_2[Y]$

- 理解：相同的  $X$  取值  $\Rightarrow$  相同的  $Y$  取值

- 称为： $X$  函数确定  $Y$ ,  $Y$  函数依赖于  $X$

# 候选键：一种特殊的函数依赖

- 可以通过函数依赖定义候选键
  - 已知 $R(U)$ 是属性集 $U = \{A_1, \dots, A_m\}$ 上的一个关系模式
- 定义： $X \subseteq U$ 是候选键，如果满足下述条件
  - $X \rightarrow U$  (X函数确定U)
  - $\forall Y \subset X, Y \nrightarrow U$  (X是最小的)
- Super key (超键)：满足上述第一个条件
  - 候选键是一种超键
  - 候选键同时满足第二个条件：它的真子集不是超键

# 函数依赖：举例

A	B	C	D
a1	b1	c1	d1
a1	b1	c1	d2
a1	b2	c2	d1
a2	b1	c3	d1

**$AB \rightarrow C$**

# 观察一下

A	B	C	D
a1	b1	c1	d1
a1	b1	c1	d2
a1	b2	c2	d1
a2	b1	c3	d1

**AB→C**

- 候选键是什么？

- 考虑包含1列的情形：A✗，B✗，C✗，D✗
- 考虑包含2列的情形：AB✗，AC✗，AD✗，BC✗，BD✗，CD✓
- 考虑包含3列的情形：ABC✗，ABD✓，ACD✗，BCD✗
- 上述函数依赖中AB不是候选键，（找候选键复杂性可以是指数级的）

- A→C? ✗

- A→A? ✓

- ABD→A? ✓

# 可以简化一下吗？

- 函数依赖的分解和合并
- 目标：属性集 $X \rightarrow$ 单个属性 $A$

# 函数依赖的推导

- Armstrong规则

- 自反律: 如果  $Y \subseteq X$ , 那么  $X \rightarrow Y$
- 增补律: 如果  $X \rightarrow Y$ , 那么任取  $Z$ ,  $XZ \rightarrow YZ$
- 传递律: 如果  $X \rightarrow Y$  而且  $Y \rightarrow Z$ , 那么  $X \rightarrow Z$

- 可以得到

- 分解: 如果  $X \rightarrow YZ$ , 那么  $X \rightarrow Y$  而且  $X \rightarrow Z$
- 合并: 如果  $X \rightarrow Y$  而且  $X \rightarrow Z$ , 那么  $X \rightarrow YZ$

# 分解的证明

- 如果  $X \rightarrow YZ$ ，那么  $X \rightarrow Y$  而且  $X \rightarrow Z$

- 证明：

$\because X \rightarrow YZ$

$\because YZ \rightarrow Y$  (自反律)

$\therefore X \rightarrow Y$  (传递律)

同理，可以证明  $X \rightarrow Z$

证明完毕。

# 合并的证明

- 如果 $X \rightarrow Y$ 而且 $X \rightarrow Z$ , 那么 $X \rightarrow YZ$

- 证明:

$\because X \rightarrow Z$

$\therefore XX \rightarrow XZ$  (增补律)

$\therefore X \rightarrow XZ$  ( $XX$ 就是 $X$ ) (1)

$\because X \rightarrow Y$

$\therefore XZ \rightarrow YZ$  (增补律) (2)

由(1)(2),  $X \rightarrow YZ$  (传递律)

证明完毕。



# 属性集 $X \rightarrow$ 单个属性 $A$

- $X \rightarrow YZ \iff X \rightarrow Y$  而且  $X \rightarrow Z$ 
  - 合并和分解是互为可逆的
  - 只要找出对于所有单一属性的函数依赖关系
  - 就可以得到所有的函数依赖关系
- 只需考虑 “属性集 $X \rightarrow$ 单个属性 $A$ ” 形式的函数依赖
- ☞ 已知一组函数依赖，怎样找出所有的函数依赖？

# 函数依赖的闭包

- 函数依赖集F的闭包:  $F^+$ 
  - 从F可以推导出的所有函数依赖的集合
  - 算法
    - 反复使用Armstrong规则
    - 直到不增加新的函数依赖为止

# 举例：函数依赖的闭包

Contracts(contractid, supplierid, projectid,  
deptid, partid, quantity, value)

- 我们用一个字母来简记各列CSJDPQV
  - 假设已知下述函数依赖关系
    - 其中C是主键，所以 $C \rightarrow SJDPQV$
    - 一个项目买一种零件只用一个合同： $JP \rightarrow C$
    - 一个部门从一个供货商至多购买一种零件： $DS \rightarrow P$
  - 已知 $F = \{C \rightarrow SJDPQV, JP \rightarrow C, DS \rightarrow P\}$
- ☞ 求函数依赖集的闭包 $F^+$

# 重新想一下：自反律？

- 自反律

- 如果  $Y \subseteq X$ ，那么  $X \rightarrow Y$

- 假设  $X$  中有  $k$  个属性，那么有多少种可能的自反律规律？

- 换言之， $X$  有多少个不同的非空子集？

- 指数个： $2^k - 1$

- 假设关系模式中有  $m$  个属性，那么有多少种可能的自反律？

- 包含1个属性的集合  $X$  有  $\binom{m}{1}$  个，有  $2^1 - 1$  种自反律

- ...

- 包含  $K$  个属性的集合  $X$  有  $\binom{m}{k}$  个，有  $2^k - 1$  种自反律

- ...

- 总计有： $\sum_{k=1}^m \binom{m}{k} (2^k - 1)$

- 列举自反律产生的函数依赖：费力而没有什么意义

# 重新想一下：增补律？

- 增补律

- 如果  $X \rightarrow Y$ ，那么任取  $Z$ ， $XZ \rightarrow YZ$

- 假设除去  $X$  和  $Y$  之外，还有  $l$  个属性，那么有多少种可能的增补律规律？

- 从  $l$  个属性中组成  $Z$  有多少中情况？

- 指数个： $2^l - 1$

- 列举增补律产生的函数依赖：费力而没有什么意义

# 重新想一下：传递律？

- 传递律

- 如果  $X \rightarrow Y$  而且  $Y \rightarrow Z$ ，那么  $X \rightarrow Z$
- 相对于自反律和增补律而言，很少！

- 所以，求闭包重点是使用传递律

- 什么情况下使用增补律和自反律呢？

- 为了创造条件可以使用传递律
- 使右侧为全部属性，即左侧是候选键

# 传递律产生的规律？

- 已知  $F = \{C \rightarrow SJDPQV, JP \rightarrow C, DS \rightarrow P\}$

求函数依赖集的闭包  $F^+$

- $C \rightarrow SJDPQV$ ，所以  $C \rightarrow CSJDPQV$ （增补律）
- $JP \rightarrow C$ ，所以  $JP \rightarrow CSJDPQV$ （传递律）
- $DS \rightarrow P$ ，所以  $JDS \rightarrow JP$ （增补律）  
所以  $JDS \rightarrow CSJDPQV$ （传递律）
- 所以，这里  $C$ 、 $JP$ 、 $JDS$  都是候选键

# 对单个属性的函数依赖

- 已知  $F = \{C \rightarrow SJDPQV, JP \rightarrow C, DS \rightarrow P\}$
- $C \rightarrow CSJDPQV, JP \rightarrow CSJDPQV, JDS \rightarrow CSJDPQV$
- 分解可以得到
  - $C \rightarrow S, C \rightarrow J, C \rightarrow D, C \rightarrow P, C \rightarrow Q, C \rightarrow V$
  - $JP \rightarrow C, JP \rightarrow S, JP \rightarrow J, JP \rightarrow D, JP \rightarrow P, JP \rightarrow Q, JP \rightarrow V$
  - $JDS \rightarrow C, JDS \rightarrow S, JDS \rightarrow J, JDS \rightarrow D, JDS \rightarrow P, JDS \rightarrow Q, JDS \rightarrow V$
- 当然还有自反律和增补律可以产生的大量规律



# 属性集的闭包

- 函数依赖F的闭包： $F^+$ 
  - 根据F可以推导出的所有函数依赖
- 属性集X的闭包： $X^+$ 
  - 相对于一个函数依赖集F
  - 计算属性集X函数确定的所有属性
  - $X \rightarrow X^+$

# 举例

- 已知  $F = \{C \rightarrow SJDPQV, JP \rightarrow C, DS \rightarrow P\}$
- 求JDS属性的闭包？
  - 在F下，由JDS函数确定的所有属性？

# 属性闭包算法

closure = X;

do {

    changed = false;

    if (存在( $Y \rightarrow Z \in F$ ) and  
        ( $Y \subseteq \text{closure}$ ) and  $!(Z \subseteq \text{closure})$ )

        then { closure = closure  $\cup$  Z; changed= true;}

} while (changed);

最后closure包含X函数确定的所有属性

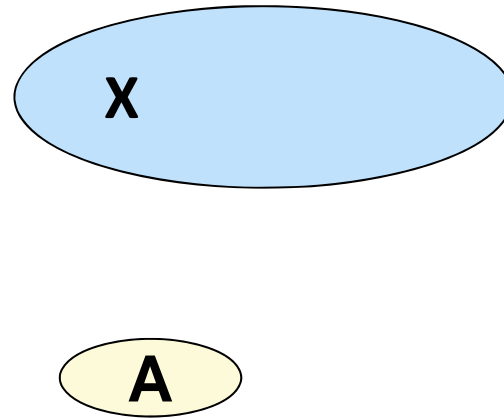
# 举例

- 已知  $F = \{C \rightarrow SJDPQV, JP \rightarrow C, DS \rightarrow P\}$
- 求  $JDS$  属性的闭包？
- 初始化,  $\text{Closure} = \{J, D, S\}$
- $DS \rightarrow P$  且  $DS \subseteq \text{Closure}$ ,  $\text{Closure} = \{S, J, D, P\}$
- $JP \rightarrow C$  且  $JP \subseteq \text{Closure}$ ,  $\text{Closure} = \{C, S, J, D, P\}$
- $C \rightarrow SJDPQV$  且  $C \subseteq \text{Closure}$ ,  $\text{Closure} = \{C, S, J, D, P, Q, V\}$

# 函数依赖

- 定义
- 规律
  - 自反律, 增补律, 传递律
  - 分解, 合并
- 闭包
  - 函数依赖的闭包
  - 属性的闭包
- 函数依赖与2NF,3NF,BCNF的关系

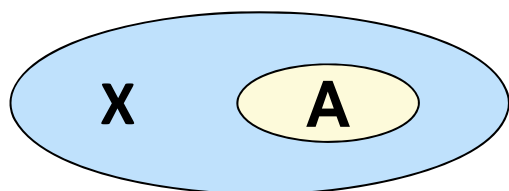
可能有什么样的函数依赖呢？  
“属性集 $X \rightarrow$ 单个属性 $A$ ”



分情况讨论 $X$ 和 $A$

# 类型1：平凡依赖

## “属性集 $X \rightarrow$ 单个属性 $A$ ”



平凡依赖✓

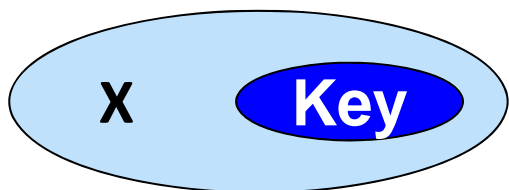
$\{A\} \subseteq X$ , 有 $X \rightarrow A$

这种函数依赖是正常的😊  
大量存在的

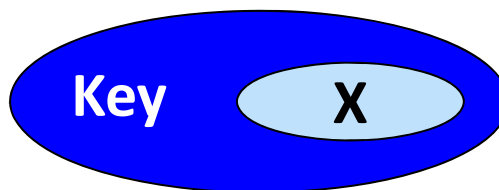
没有数据冗余问题

👉 接下来，我们考虑非平凡的函数依赖  
从 $X$ 和 $A$ 与候选键的关系来分类讨论

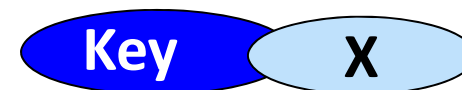
# “ $X \rightarrow A$ ”与键的关系有下述几种可能



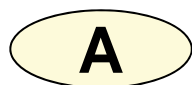
X是超键



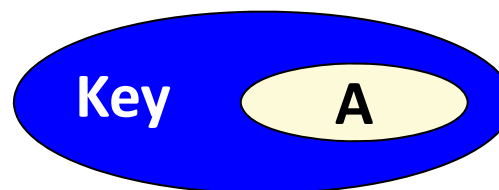
X是候选键的一部分



X包含非键的属性，  
并且X不是超键



A是非键属性

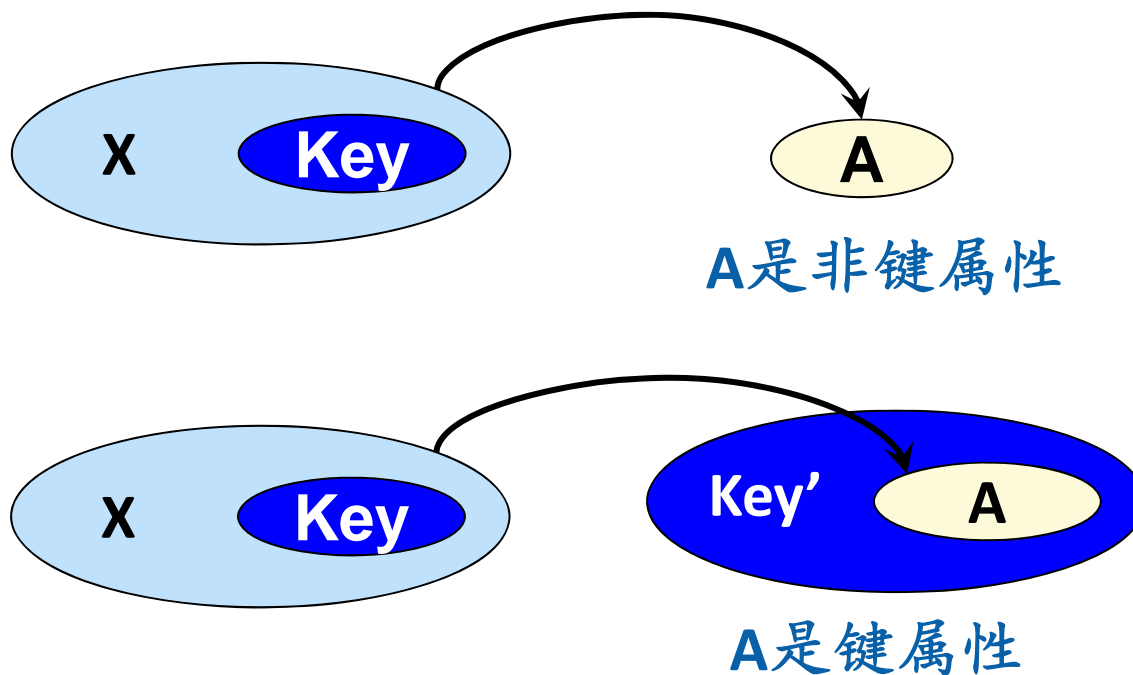


A是键属性

所以，有 $3 \times 2 = 6$ 种组合



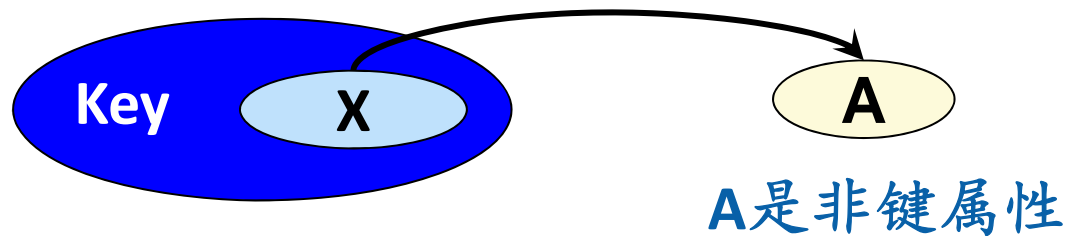
## 类型2: X是超键✓



- X是超键（包括X是候选键的情形）
- X当然可以函数确定任何属性
- 正常的😊，没有数据冗余问题

## 类型3：部分依赖

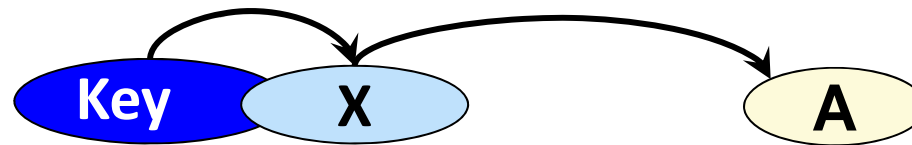
**X**是候选键的一部分，**A**是非键属性



- 这会导致冗余
- 例如：  
create table *Reserves*(*sid* integer, *bid* integer, *day* date,  
                          *credit\_card* char(16) );
  - 主键是sid, bid, day的组合
  - 而假设已知sid ->credit\_card

## 类型4：非键传递依赖

**X**包含非键的属性，**X**不是超键，**A**是非键属性



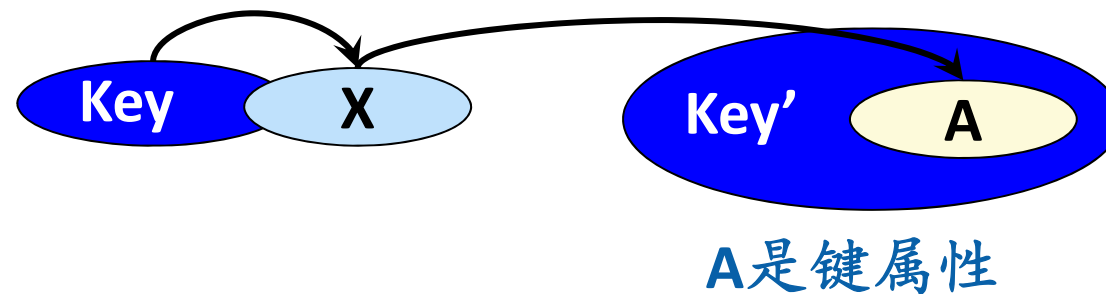
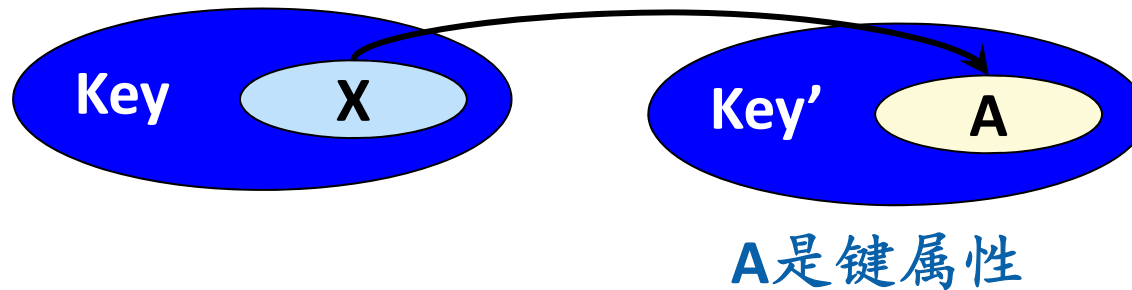
**A**是非键属性

- 这会导致冗余
- 在前面例子的Employee表中
  - ❑ 主键为ID
  - ❑ 但是Rating -> Wage

**Employee**

ID	Name	Rating	Wage
1	Alan	5	30
2	Bob	7	50
3	Carol	7	50
4	Dan	5	30

## 类型5：对于键属性的函数依赖

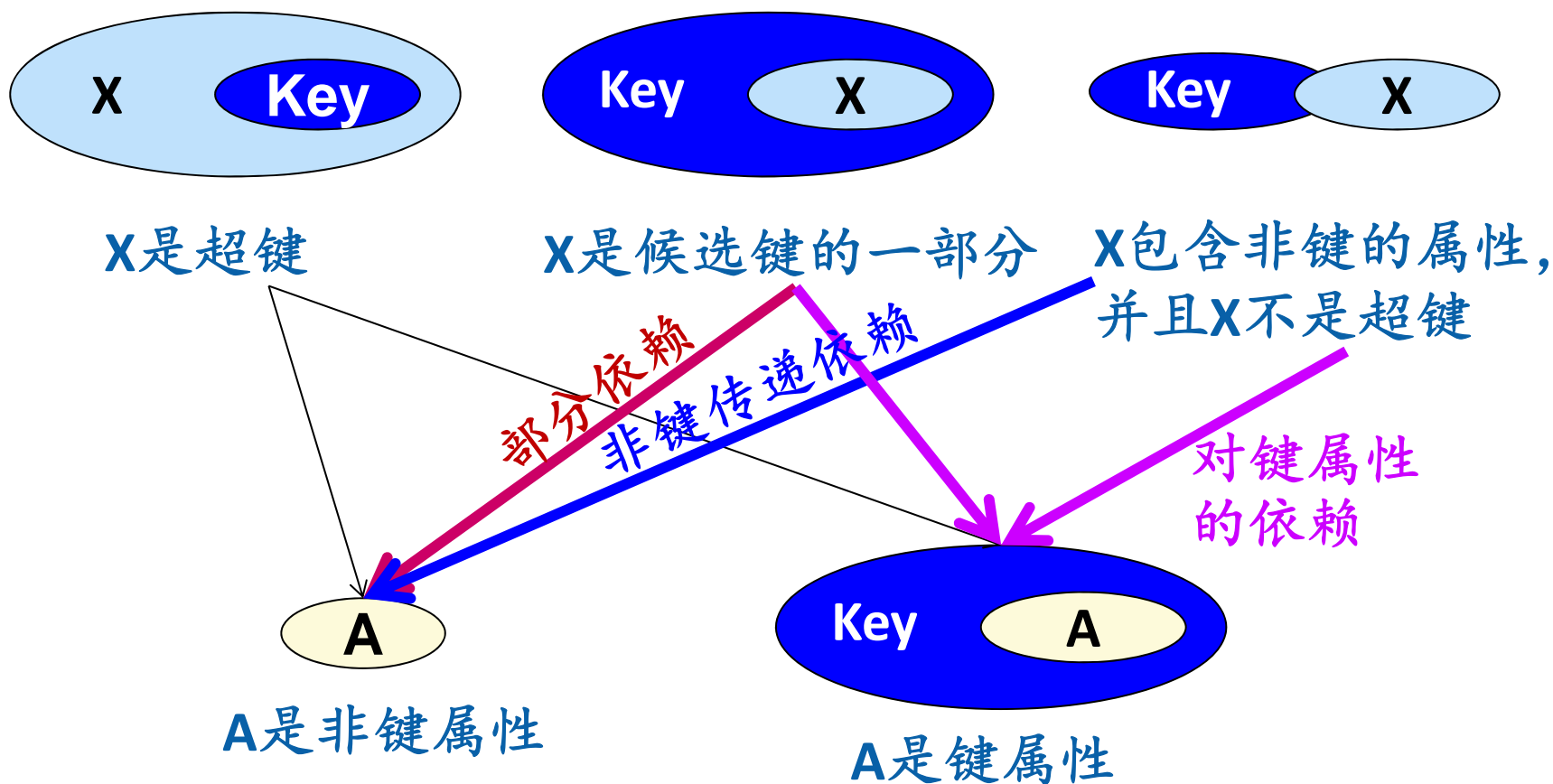


- 这些都可能产生冗余
- 一个例子：
  - ABD和CD是候选键

A	B	C	D
a1	b1	c1	d1
a1	b1	c1	d2
a1	b2	c2	d1
a2	b1	c3	d1

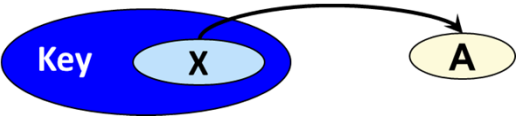
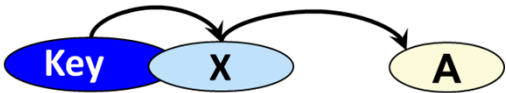
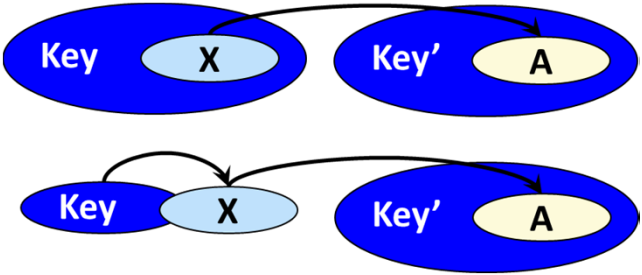
**AB→C**

# “ $X \rightarrow A$ ”与键的关系有下述几种可能



所以, 有 $3 \times 2 = 6$ 种组合

# 函数依赖类型与消除依赖的范式

		2NF	3NF	BCNF
1	平凡依赖	✓	✓	✓
2	X是超键	✓	✓	✓
3	部分依赖 	消除	消除	消除
4	非键传递依赖 		消除	消除
5	对于键属性的函数依赖 			消除

# 具体而言

- 什么是满足1NF的模式？

- 所有属性（列）都是原子类型，这样的模式满足1NF
- 关系模型的基本要求

- 什么是满足2NF的模式？

- 满足1NF，并且没有部分依赖，这样的模式满足2NF

- 什么是满足3NF的模式？

- 满足2NF，并且没有非键传递依赖，这样的模式满足3NF

- 什么是满足BCNF的模式？

- 满足3NF，并且没有对键属性的函数依赖，这样的模式满足BCNF

# 回到开始的范式关系图

**1NF:** 所有属性（列）都是原子类型

**2NF:** 消除函数依赖中的部分依赖

**3NF:** 消除函数依赖中的非键传递依赖

**BCNF:** 消除所有函数依赖（希望达到）



# 那么如何判断一个关系模式是否满足1NF, 2NF, 3NF, BCNF?

- 找出所有的函数依赖关系
- 然后看看其中是否包含
  1. 存在部分依赖?
    - a) Yes: 只满足1NF, 完成
    - b) No: 满足2NF, 继续
  2. 存在非键传递依赖?
    - a) Yes: 完成
    - b) No: 满足3NF, 继续
  3. 存在对于键属性的函数依赖?
    - a) Yes: 完成
    - b) No: 满足BCNF, 完成

# 举例

Contracts(contractid, supplierid, p<sup>r</sup>oj<sup>e</sup>ctid,  
deptid, p<sup>a</sup>rtid, q<sup>t</sup>y, v<sup>a</sup>lue)

- 我们用一个字母来简记各列CSJDPQV
- 函数依赖关系  $F = \{C \rightarrow SJDPQV, JP \rightarrow C, DS \rightarrow P\}$
- 满足什么范式?

# 解答

Contracts(contractid, supplierid, projectid,  
deptid, partid, qty, value)

- 我们用一个字母来简记各列CSJDPQV
- 函数依赖关系  $F = \{C \rightarrow SJDPQV, JP \rightarrow C, DS \rightarrow P\}$
- 候选键为：C, JP, JDS；非键属性：Q, V
  1. 存在部分依赖？
    - a) No：满足2NF，继续
  2. 存在非键传递依赖？
    - a) No：满足3NF，继续
  3. 存在对于键属性的函数依赖？
    - a) Yes：完成