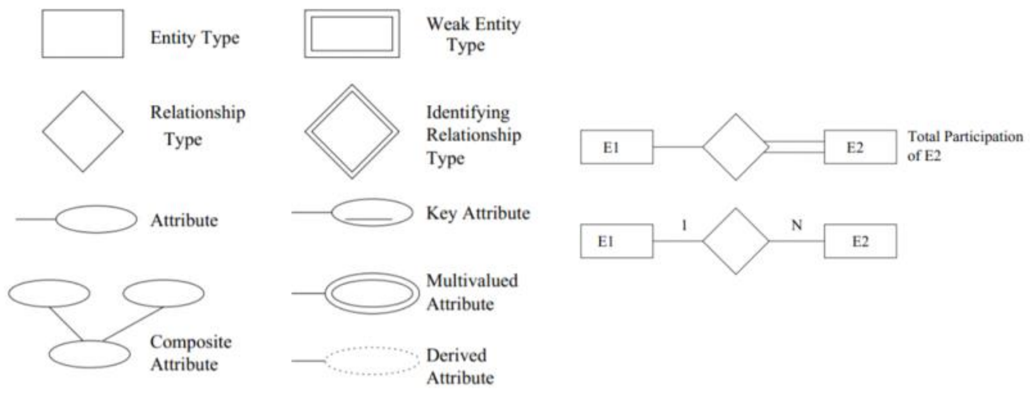
ER model

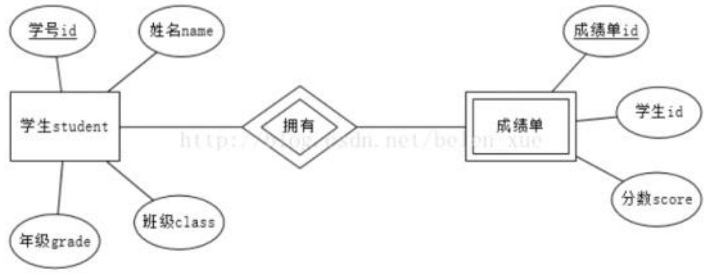
Entities represent things in the real world.

Attributes describe properties of entities.

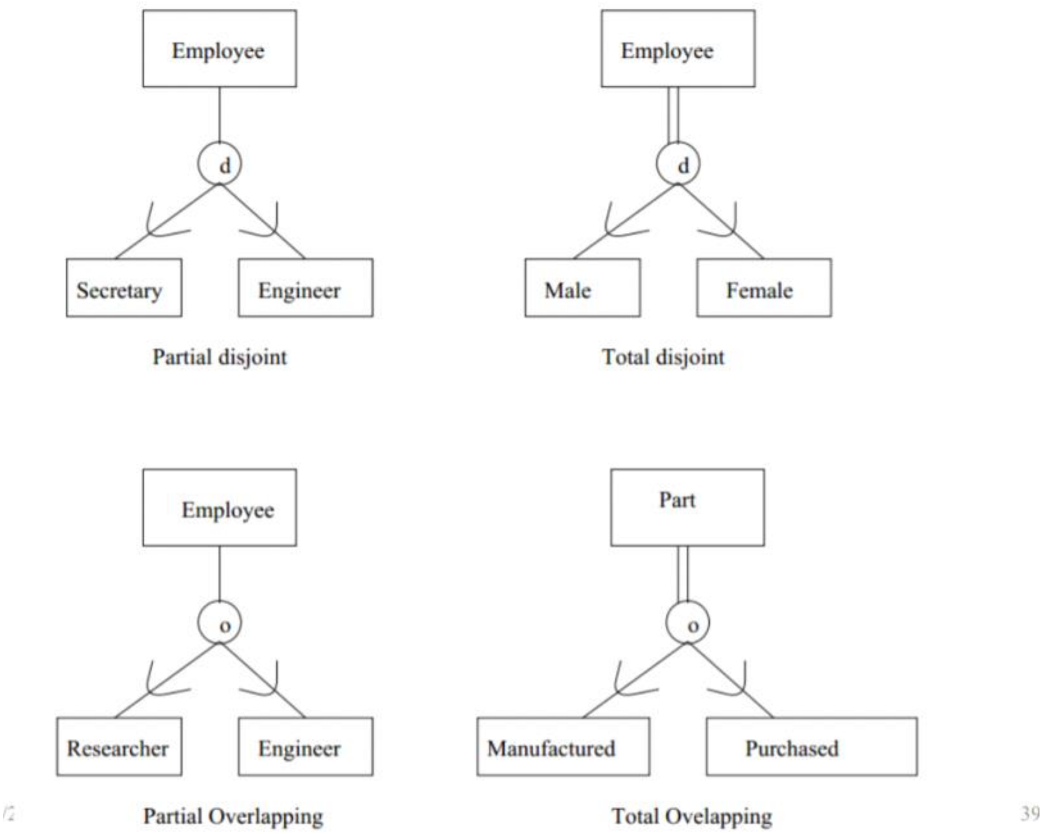
A relationship represents an association between things.




一个实体必须依赖于另一个实体存在，那么前者是弱实体，后者是强实体，弱实体必须依赖强实体存在，例如上图的学生实体和成绩单实体，成绩单依赖于学生实体而存在，因此学生是强实体，而成绩单是弱实体, 与弱实体的联系用双线菱形框表示，例如：



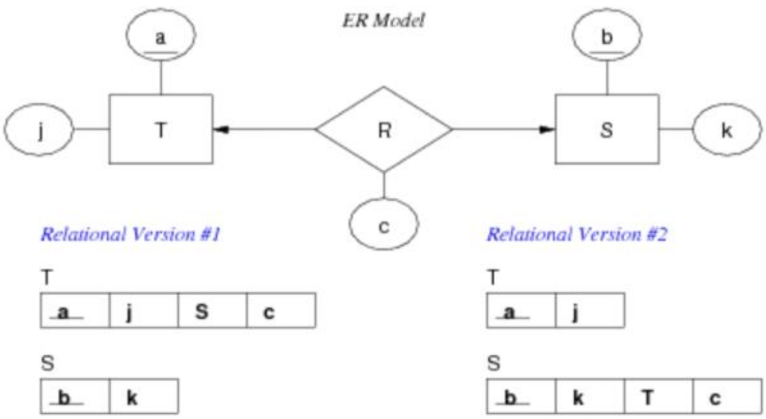

Disjoint and overlapping



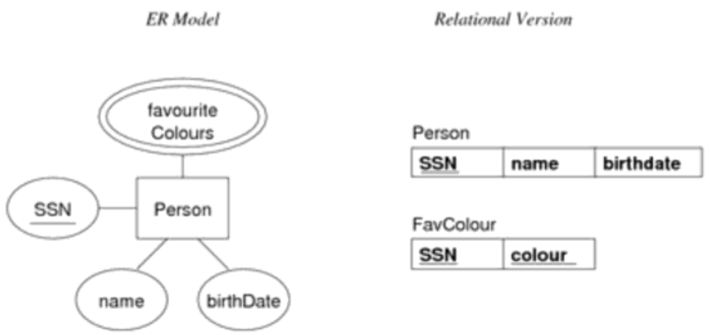

RDB:

Key constraint: 侯选键必须唯一
Entity integrity：所有主键的属性不能为nulll
Referential integrity:外键要么为空，要么为参照关系的主键的值
插入数据： Key constraint 是否唯一
Referential integrity 不为null
Entity integrity 外键必须有对应的值
删除数据： Referential integrity 当前主键是其他表的外键

ER -> RDB mapping
1.实体 E
Attributes: 所有单一属性，除了多值属性
Key：选主键(可以是一个或多个)
2.弱实体 W
所有单一属性
key: Foreign Key（W依赖的实体E的主键） + partial key（W的主键
3.找1:1关系（实体S、T）
关系带有的属性加在实体T（如果T完全参与）后面，（如果没有任何一个实体是完全参与，则可以选任何一个实体进行添加）
在T的关系后写一个S的主键作为外键（**箭号指向S的主键**）

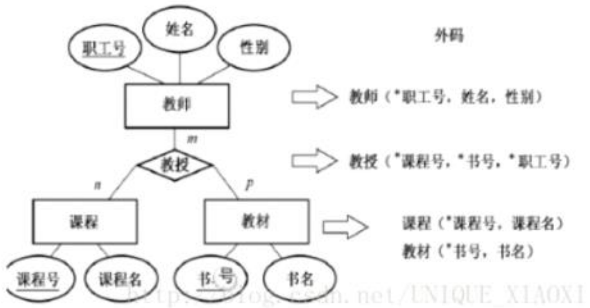



4.找1：N关系
把1这侧的entity的主键作为外键写在N这侧实体后面（**箭号指向1的主键**）
5.找M:N关系
新建一个表
把S、T的主键都作为新表的外键（**两个箭号指向S、T的主键**）
**两个都要有下划线**



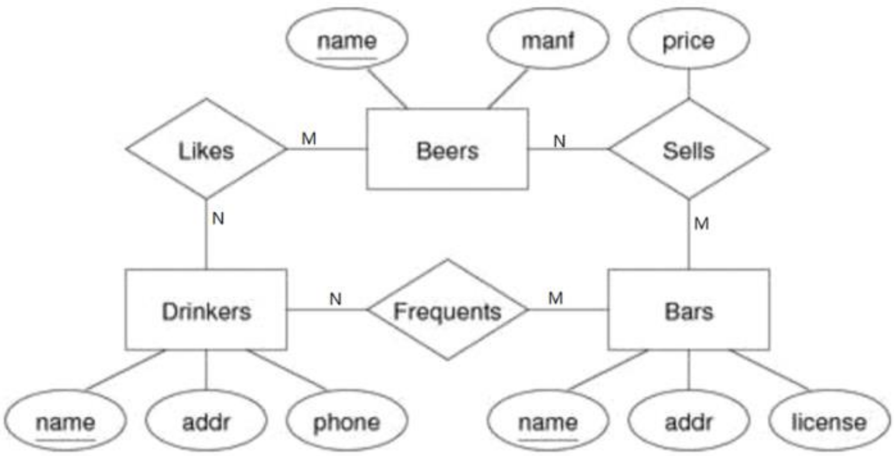

6.多值属性
新建一个表
将该属性隶属于的实体E的主键作为外键（**箭号指向E的主键**）

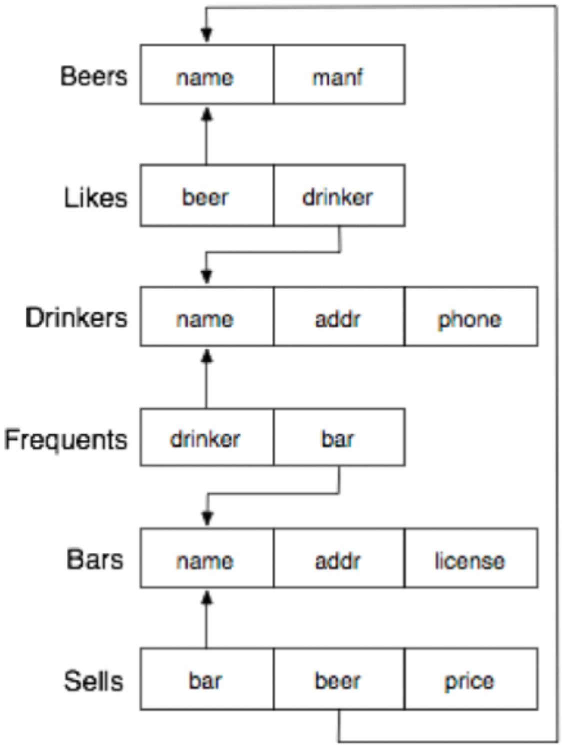
7.第七步 如果又多个关系，一个菱形联系 连接多个实体




PA：

ER design for Beer database:




RA

$$R1 \leftarrow ENROLMENT \times RESEARCHER$$

**箭头可以把结果存到新表中**

**Select**：

$$\sigma_{(Supervisor=1)\text{AND NOT}(Name\neq\text{"PH.D"})}(ENROLMENT)$$

using AND, OR and NOT.

**Project:**

$$\pi_{Department,Name}(ENROLMENT)$$

选几列的结果并去重

| A | B | D |
|---|---|---|
| a | 1 | 4 |
| b | 2 | 5 |
| c | 4 | 4 |
| d | 8 | 5 |
| e | 1 | 4 |
| f | 2 | 5 |

| B | D |
|---|---|
| 1 | 4 |
| 2 | 5 |
| 4 | 4 |
| 8 | 5 |

**Union:** 并集

**Intersection:** 交集

**Difference:** 差集

s1 = Sel [B = 1] (r1)

| A | B | C | D |
|---|---|---|---|
| a | 1 | x | 4 |
| e | 1 | y | 4 |

s2 = Sel [C = x] (r1)

| A | B | C | D |
|---|---|---|---|
| a | 1 | x | 4 |
| d | 8 | x | 5 |

s1 - s2

| A | B | C | D |
|---|---|---|---|
| e | 1 | y | 4 |

s2 - s1

| A | B | C | D |
|---|---|---|---|
| d | 8 | x | 5 |

**Cartesian Product**

R

| A | B | C |
|---|---|---|
| a1 | b1 | c1 |
| a1 | b2 | c2 |
| a2 | b2 | c1 |

S

| A | B | C |
|---|---|---|
| a1 | b2 | c2 |
| a1 | b3 | c2 |
| a2 | b2 | c1 |

$R \times S$

| A | B | C | A | B | C |
|---|---|---|---|---|---|
| a1 | b1 | c1 | a1 | b2 | c2 |
| a1 | b1 | c1 | a1 | b3 | c2 |
| a1 | b1 | c1 | a2 | b2 | c1 |
| a1 | b2 | c2 | a1 | b2 | c2 |
| a1 | b2 | c2 | a1 | b3 | c2 |
| a1 | b2 | c2 | a2 | b2 | c1 |
| a2 | b2 | c1 | a1 | b2 | c2 |
| a2 | b2 | c1 | a1 | b3 | c2 |
| a2 | b2 | c1 | a2 | b2 | c1 |

An Introduction to Database System

下面为关系R和关系S 两张表:

**关系 R**

| A | B | C |
|---|---|---|
| 1 | 2 | 3 |
| 4 | 5 | 6 |
| 7 | 8 | 9 |

**关系 S**

| A | B | C |
|---|---|---|
| 2 | 4 | 6 |
| 4 | 5 | 6 |

(1) 并运算

**R∪S**

| A | B | C |
|---|---|---|
| 1 | 2 | 3 |
| 4 | 5 | 6 |
| 7 | 8 | 9 |
| 2 | 4 | 6 |

（2）差运算

**R-S**

| A | B | C |
|---|---|---|
| 1 | 2 | 3 |
| 7 | 8 | 9 |

（3）笛卡尔积运算

**R×S**

| R.A | R.B | R.C | S.A | S.B | S.C |
|---|---|---|---|---|---|
| 1 | 2 | 3 | 2 | 4 | 6 |
| 1 | 2 | 3 | 4 | 5 | 6 |
| 4 | 5 | 6 | 2 | 4 | 6 |
| 4 | 5 | 6 | 4 | 5 | 6 |
| 7 | 8 | 9 | 2 | 4 | 3 |
| 7 | 8 | 9 | 4 | 5 | 6 |

（5）投影

$\Pi_{C,A}(R)$

| C | A |
|---|---|
| 3 | 1 |
| 6 | 4 |
| 9 | 7 |

（6）选择

$\sigma_{B>'4'}(R)$

| A | B | C |
|---|---|---|
| 4 | 5 | 6 |
| 7 | 8 | 9 |

Join:

Natural join

$$ENROLMENT \bowtie_{(Supervisor),(Person\#)} RESEARCHER$$

$$ENROLMENT \bowtie_{(Department,Name),(Department,Name)} COURSE$$

举例：有关系R和关系S两张表

R

| A | B | C |
|---|---|---|
| 2 | 4 | 6 |
| 3 | 5 | 7 |
| 7 | 4 | 6 |
| 5 | 4 | 7 |

S

| B | C | D |
|---|---|---|
| 5 | 7 | 3 |
| 4 | 6 | 2 |
| 5 | 7 | 9 |
| 5 | 6 | 3 |

http://blog.csdn.net/mingxuanyun

下图表示关系R和S的自然连接:

$R \bowtie S$

| A | B | C | D |
|---|---|---|---|
| 2 | 4 | 6 | 2 |
| 3 | 5 | 7 | 3 |
| 3 | 5 | 7 | 9 |
| 7 | 4 | 6 | 2 |

theta-join

$$r \bowtie_B s = \{t_1 || t_2 : t_1 \in r \text{ and } t_2 \in s \text{ and } B\}$$

| A | B | C |
|---|---|---|
| 1 | 2 | 3 |
| 4 | 5 | 6 |
| 7 | 8 | 9 |

关系R

| D | E |
|---|---|
| 3 | 1 |
| 6 | 2 |

关系S

$R \bowtie_{B<D} S$

| A | B | C | D | E |
|---|---|---|---|---|
| 1 | 2 | 3 | 3 | 1 |
| 1 | 2 | 3 | 6 | 2 |
| 4 | 5 | 6 | 6 | 2 |

$R \bowtie_{B<D} S$

Equi-join

$$\boldsymbol{EMPNAMES} \bowtie_{(Ssn=Essn)} \boldsymbol{DEPENDENT}$$

Divide:

$$R \div S = \{t : t \times S \subseteq \mathrm{R}\}$$

| P | |
|---|---|
| A | B |
| $a_1$ | $b_1$ |
| $a_1$ | $b_2$ |
| $a_2$ | $b_1$ |
| $a_3$ | $b_2$ |
| $a_4$ | $b_1$ |
| $a_5$ | $b_1$ |
| $a_5$ | $b_2$ |

| Q |
|---|
| B |
| $b_1$ |
| $b_2$ |

$P \div Q =$

| A |
|---|
| $a_1$ |
| $a_5$ |

Prac ans:

已知一个关系数据库的模式如下:

S (SNO,SNAME,SCITY)

P (PNO,PNAME,COLOR,WEIGHT)

J (JNO,JNAME,JCITY)

SPJ (SNO,PNO,JNO,QTY)

供应商S由供应商代码SNO、供应商姓名SNAME、供应商所在城市SCITY组成;零件P由零件代码PNO、零件名PNAME、颜色COLOR、重量WEIGHT组成;工程项目J由工程项目代码JNO、工程项目名JNAME、和所在城市JCITY组成;供应情况SPJ由供应

商代码SNO、零件代码PNO、工程项目代码JNO、供应数量QTY组成。

用关系代数表达式表示下面的查询要求：

（1）找出向北京的供应商购买重量大于30的零件工程名。

（2）求供应工程J1零件的供应商代码

（3）求供应工程J1零件P1的供应上代码

（4）求供应工程J1零件为红色的供应商代码

（5）求没有使用天津供应商生产的红色零件的工程项目代码

（6）求至少用了供应商S1所供应的全部零件的工程项目代码

(1)

$$\pi_{JNAME}(\sigma_{WEIGHT > 30 \wedge SCITY = '北京'}(S \bowtie P \bowtie J \bowtie SPJ))$$

(2)

$$\pi_{SNO}(\sigma_{JNO = 'J1'}(SPJ))$$

(3)

$$\pi_{SNO}(\sigma_{JNO = 'J1' \wedge PNO = 'P1'}(SPJ))$$

(4)

$$\pi_{SNO}(\sigma_{JNO = 'J1' \wedge COLOR = '红色'}(SPJ \bowtie P))$$

(5)

$$\pi_{JNO}(J) - \pi_{JNO}(\sigma_{SCITY = '天津' \wedge COLOR = '红色'}(SPJ \bowtie S \bowtie P))$$

(6)

$$\pi_{PNO, JNO}(SPJ) \div \pi_{PNO}(\sigma_{SNO = 'S1'}(SPJ))$$

Sql
https://www.tutorialspoint.com/sql/sql-expressions.htm
Data types:
char(n) a fixed length string
varchar(n) a variable length string
date Format: YYYY-MM-DD
Integer
Text maximum length of 65,535

常见的数字操作:

– AVG(attr) ... mean of values for attr
– COUNT(attr) ... number of rows in attr column
– MIN/MAX(attr) ... min/max of values for attr
– SUM(attr) ... sum of values for attr

RDD
RDB 概念

候选键：能惟一标识元组，并且不含多余属性的属性(组合属性)
主键：从若干个候选键中指定一个作为主键
超键：除可以包含一个候选键外，还可以包含其它属性

例子：

学生（学号，姓名，性别，身份证号）

超键：

由超键的定义可知，学生表中含有学号或者身份证号的任意组合都为此表的超键。
如：（学号）、（学号，姓名）

候选键：

候选键属于超键，它是最小的超键，就是说如果再去掉候选键中的任何一个属性它就不再是超键了。学生表中的候选键为：（学号）、（身份证号）。

主键：

主键就是候选键里面的一个，是人为规定的，我们通常会让“学号”做主键，

FD:
定义:设X,Y是关系R的两个属性集合，当任何时刻R中的任意两个元组中的X属性值相同时，则它们的Y属性值也相同，则称X函数决定Y，或Y函数依赖于X。
例子:在设计学生表时，一个学生的学号能决定学生的姓名，也可称姓名属性依赖于学号.

部分函数依赖：设X,Y是关系R的两个属性集合，存在X→Y，若X’是X的真子集，存在X’→Y，则称Y部分函数依赖于X。

举个例子：学生基本信息表R中（学号，身份证号，姓名）当然学号属性取值是唯一的，在R关系中，（学号，身份证号）->（姓名），（学号）->（姓名），（身份证号）->（姓名）；所以姓名部分函数依赖与（学号，身份证号）；

完全函数依赖：设X,Y是关系R的两个属性集合，X’是X的真子集，存在X→Y，但对每一个X’都有X’!→Y，则称Y完全函数依赖于X。

例子：学生基本信息表R（学号，班级，姓名）假设不同的班级学号有相同的，班级内学号不能相同，在R关系中，（学号，班级）->（姓名），但是（学号）->(姓名)不成立，（班级）->(姓名)不成立，所以姓名完全函数依赖与（学号，班级）；

传递函数依赖：设X,Y,Z是关系R中互不相同的属性集合，存在X→Y(Y !→X),Y→Z，则称Z传递函数依赖于X。

例子：在关系R(学号 ,宿舍, 费用)中，(学号)->(宿舍),宿舍！=学号，(宿舍)->(费用)，所以符合传递函数的要求；

Armstrong’s axioms

F1. Reflexivity  e.g.  $X \to X$

- a formal statement of trivial dependencies; useful for derivations

F2. Augmentation  e.g.  $X \to Y \Rightarrow XZ \to YZ$

- if a dependency holds, then we can freely expand its left hand side

F3. Transitivity  e.g.  $X \to Y,\ Y \to Z \Rightarrow X \to Z$

- the "most powerful" inference rule; useful in multi-step derivations

F4. Additivity e.g. X → Y, X → Z ⇒ X → YZ

- useful for constructing new right hand sides of fds (also called union)

F5. Projectivity e.g. X → YZ ⇒ X → Y, X → Z

- useful for reducing right hand sides of fds (also called decomposition)

F6. Pseudotransitivity e.g. X → Y, YZ → W ⇒ XZ → W

- shorthand for a common transitivity derivation

闭包：闭包就是由一个属性直接或间接推导出的所有属性的集合

- $f = \{a \to b, b \to c, a \to d, e \to f\}$
- 则a的闭包就是$\{a, b, c, d\}$

NF:

**1、第一范式：1NF(First Normal Form) 每个属性都不可再分**

取消表中套表的现象。以下表中套表，不符合1NF,不符合关系数据库。

| name | tel | | age |
|---|---|---|---|
| 大宝 | 13612345678 | | 22 |
| 小明 | 13988776655 | 010 - 1234567 | 21 |

修改为（name,tel,age）和（tel，cellphone，fix_phone）

**2、第二范式：2NF（Second Normal Form)**

简单的说，第二范式就是在符合1NF的基础上，**非主属性(non-primary attribute)完全**函数依赖(full functional dependency)于**主属性(primary attribute)**。

| 学生 | 课程 | 老师 | 老师职称 | 教材 | 教室 | 上课时间 |
|---|---|---|---|---|---|---|
| 小明 | 一年级语文（上） | 大宝 | 副教授 | 《小学语文1》 | 101 | 14：30 |

一个学生上一门课，一定在特定某个教室。所以有（学生，课程）->教室

一个学生上一门课，一定是特定某个老师教。所以有（学生，课程）->老师

一个学生上一门课，他老师的职称可以确定。所以有（学生，课程）->老师职称

一个学生上一门课，一定是特定某个教材。所以有（学生，课程）->教材

一个学生上一门课，一定在特定时间。所以有（学生，课程）->上课时间

因此（学生，课程）是一个码。

然而，一个课程，一定指定了某个教材，一年级语文肯定用的是《小学语文1》，那么就有课程->教材。（学生，课程）是个码，课程却决定了教材，这就叫做不完全依赖(partial functional dependency)，或者说部分依赖。出现这样的情况，就不满足第二范式！

（学生，课程，老师，老师职称，教室，上课时间）和（课程，教材）

**3、第三范式：3NF（Third Normal Form)**

在符合2NF的基础上，消除传递函数依赖（transitive dependency）。什么是传递函数依赖呢：如果(A,B,C),其中A为主键，B，C为非主属性，如果存在A->B->C,则称为传递函数依赖。[**C传递依赖于A，且C为非主属性**。如果要求C也可以为主属性，那么，是BCNF]

如上例，（学生，课程，老师，老师职称，教室，上课时间）和（课程，教材）已经能够符合2NF，但是

即存在（学生，课程）->(老师)->(老师职称)的传递函数依赖关系。

会产生如下问题：

（1）老师升级了，变教授了，要改数据库，表中有N条，改了N次......（修改异常）

（2） 没人选这个老师的课了，老师的职称也没了记录……（删除异常）

（3）新来一个老师，还没分配教什么课，他的职称记到哪？ ……（插入异常）

因此修改为：

（<u>学生，课程</u>，老师，教室，上课时间）和（<u>课程</u>，教材）和（<u>老师</u>，老师职称）

**3、BC范式：BCNF（Boyce-Codd Normal Form）**

符合3NF基础上，并且，**主属性**不依赖于主属性。也可以表述如下：

若关系模式属于第一范式，且**每个属性（主属性和非主属性）**都不传递依赖于键码，则R属于BC范式。

以上两者等价。**BC范式既检查非主属性，又检查主属性。当只检查非主属性时，就成了第三范式。满足BC范式的关系都必然满足第三范式。**

lossless join decomposition, 即分解后仍可以复原原来的关系

Lossless Join Decomposition:
检查无损连接分解：

画表

*Example 5:* $R = (A,B,C,D,E,G)$,

$F = \{AB \rightarrow G,\ C \rightarrow DE,\ A \rightarrow B,\}$.

Let $R_1 = (A,B,)$, $R_2 = (C, D, E)$ and $R_3 = (A, C, G)$.

| | A | B | C | D | E | G |
|---|---|---|---|---|---|---|
| $R_1$ | a | a | b | b | b | b |
| $R_2$ | b | b | a | a | a | b |
| $R_3$ | a | b | a | b | b | a |

| | A | B | C | D | E | G |
|---|---|---|---|---|---|---|
| $R_1$ | a | a | b | b | b | b |
| $R_2$ | b | b | a | a | a | b |
| $R_3$ | a | ~~b~~ a | a | a | a | a |

2018/4/6 14

关系模式 R为 R(H,I,J,K,L)，R上的一个函数依赖集为 F={H→J,J→K,I→J,JL→H}
p={HIL，IKL，IJL}

我们列出选项B(分解成三个关系模式R1(HIL)、R2(IKL)、R3(IJL) )的初始表如表3所示：
表3选项B的初始表

| | H | I | J | K | L |
|---|---|---|---|---|---|
| HIL | a1 | a2 | b13 | b14 | a5 |
| IKL | b21 | a2 | b23 | a4 | a5 |
| IJL | b31 | a2 | a3 | b34 | a5 |

对于函数依赖集中的H→J、J→K对表3进行处理，由于属性列H和属性列J上无相同的元素，所以无法修改。但对于I→J在属性列I上对应的1、2、3行上全为a2元素，所以，将属性列J的第一行b13和第二行b23改为a3。修改后如表4所示：

表4选项B的中间表

| | H | I | J | K | L |
|---|---|---|---|---|---|
| HIL | a1 | a2 | a3 | b14 | a5 |
| IKL | b21 | a2 | a3 | a4 | a5 |
| IJL | b31 | a2 | a3 | b34 | a5 |

对于函数依赖集中的JL→H在属性列J和L上对应的1、2、3行上为a3、a5元素，所以，将属性列H的第二行b21和第三行b31改为a1。修改后如表5所示：
表5选项B的结果表

| | H | I | J | K | L |
|---|---|---|---|---|---|
| HIL | a1 | a2 | a3 | b14 | a5 |
| IKL | a1 | a2 | a3 | a4 | a5 |

| | | | | | |
|---|---|---|---|---|---|
| IJL | a1 | a2 | a3 | b34 | a5 |

从表5可以看出，第二行为a1、a2、a3、a4、a5，所以分解ρ是无损的。

Fm：最小函数依赖集

**U=ABCDEG，F={AD→E，AC→E，CB→G，BCD→AG，BD→A，AB→G,A→C}**

- 分解函数依赖的右部， F={AD→E，AC→E，BC→G，BCD→A，BCD→G，BD→A，AB→G，A→C}
- 消去左边的冗余属性（作法是属性中去掉其中的一个,看看是否依然可以推导）：F={AC→E，BC→G，BD→A，A→C}
- 消去冗余的函数依赖(做法为从F中去掉某关系,如去掉(X->Y),然后在F中求X+,如果Y在X+中,则表明x->是多余的.需要去掉.)
- ：Fm={A→E，BC→G，BD→A，A→C}

判断3NF：

左边就是候选键

左边不是候选键但右边是候选键一部分(注意这个不满足BCNF)

分解成3NF：

1.找到最小函数依赖集

2.对于Fm里面每个依赖的左侧部分,创建一个relation schemas={X ∪ A1 ∪ A2 … ∪ Am} X 是依赖的左侧，A1…Am 是依赖的右侧

3.如果没有relation schemas 包含侯选键，则为侯选键创建一个relation schemas.

*例子:*

*有关系模式R(A,B,C,D)，R上的函数依赖集F={A->C，C->A，B->AC，D->AC}*

*Fm： A->C C->A B->A D->A*

*候选键 BD*

*ρ ={AC， BA， DA， BD}*

*关系模式R<U,F>，其中U={C,T,H,R,S,G},*

*F={CS→G,C→T,TH→R,HR→C,HS→R}，将其分解成3NF并保持函数依赖。*

***最小函数依赖集为：F={CS→G,C→T,TH→R,HR→C,HS→R}***

*候选键：HS*

*ρ={R1(CSG),R2(CT),R3(THR),R4(HRC),R5(HSR)}*

判断BCNF：

所有依赖关系的左侧是superkey

i. $C \to D$, $C \to A$, $B \to C$

[hide answer]

a. Candidate keys: $B$
b. Not BCNF ... e.g. in $C \to A$, $C$ does not contain a key
c. Not 3NF ... e.g. in $C \to A$, $C$ does not contain a key, $A$ is not part of a key

ii. $B \to C$, $D \to A$

[hide answer]

a. Candidate keys: $BD$
b. Not 3NF ... neither right hand side is part of a key
c. Not BCNF ... neither left hand side contains a key

iii. $ABC \to D$, $D \to A$

[hide answer]

a. Candidate keys: $ABC$ $BCD$
b. 3NF ... $ABC \to D$ is ok, and even $D \to A$ is ok, because $A$ is a single attribute from the key
c. Not BCNF ... e.g. in $D \to A$, $D$ does not contain a key

iv. $A \to B$, $BC \to D$, $A \to C$

[hide answer]

a. Candidate keys: $A$
b. Not 3NF ... e.g. in $A \to C$, $C$ is not part of a key
c. Not BCNF ... e.g. in $BC \to D$, $BC$ does not contain a key

v. $AB \to C$, $AB \to D$, $C \to A$, $D \to B$

[hide answer]

a. Candidate keys: $AB$ $BC$ $CD$ $AD$
b. 3NF ... for $AB$ case, first two fd's are ok, and the others are also ok because the RHS is a single attribute from the key
c. Not BCNF ... e.g. in $C \to A$, $C$ does not contain a key

vi. $A \to BCD$

[hide answer]

a. Candidate keys: $A$
b. 3NF ... all left hand sides are superkeys
c. BCNF ... all left hand sides are superkeys

分解成BCNF：
1.找出一个主键
2.检查F中所有的关系，如果不符合BCNF范式，假设X->A 则分解成为s1={XA}和s2={（S-A）X }
3.对s1 和 s2 重新进行步骤2

iv. AB → D, BCD → EF, B → C

a. Candidate key: AB
b. Not BCNF, in BCD → EF, BCD does not contain a key, also in B → C, B does not contain a key
c.

We start from a schema: ABCDEF, with key AB.
The FD BCD → EF violates BCNF (FD with non key on LHS).
To fix, we need to decompose into tables: BCDEF and ABCD.
FDs for BCDEF are { BCD → EF, B → C }.
Key for BCDEF is BD, and FD B → C violates BCNF.
To fix, we need to decompose into table: BC, BDEF.
FDs for BC are { B → C} therefore key is B, therefore BCNF.
FDs for BDEF are { }, so key is BDEF and table is BCNF.
FDs for ABCD are {AB→ D, B → C }.
Key for ABCD is AB, and FD B → C violates BCNF.
To fix, we need to decompose into table: BC, ABD.
FDs for BC are { B → C} therefore key is B, therefore BCNF.
FDs for ABD are { AB→ D }, so key is AB and table is BCNF.
Final schema (with keys bold): **B**C, **AB**D, **BDEF**

iii. ABF → D, CD → E, BD → A

a. Candidate key: ABCF BCDF
b. Not BCNF, none of LHS of FDs contain a key
c.

We start from a schema: ABCDEF, with key ABCF.
The FD ABF → D violates BCNF (FD with non key on LHS).
To fix, we need to decompose into tables: ABFD and ABCEF.
FDs for ABFD are { ABF → D, BD → A}
Key for ABFD is ABF(and BDF), and FD BD → A violates BCNF.
To fix, we need to decompose into table: ABD, BDF.
FDs for ABD are { BD → A}, therefore key is BD, therefore BCNF.
FDs for BDF are { }, so key is BDF and table is BCNF.
FDs for ABCEF are {ABCF→ E} so key is ABCF and table is BCNF.
Final schema (with keys bold): A**BD** **BDF** **ABFC**E