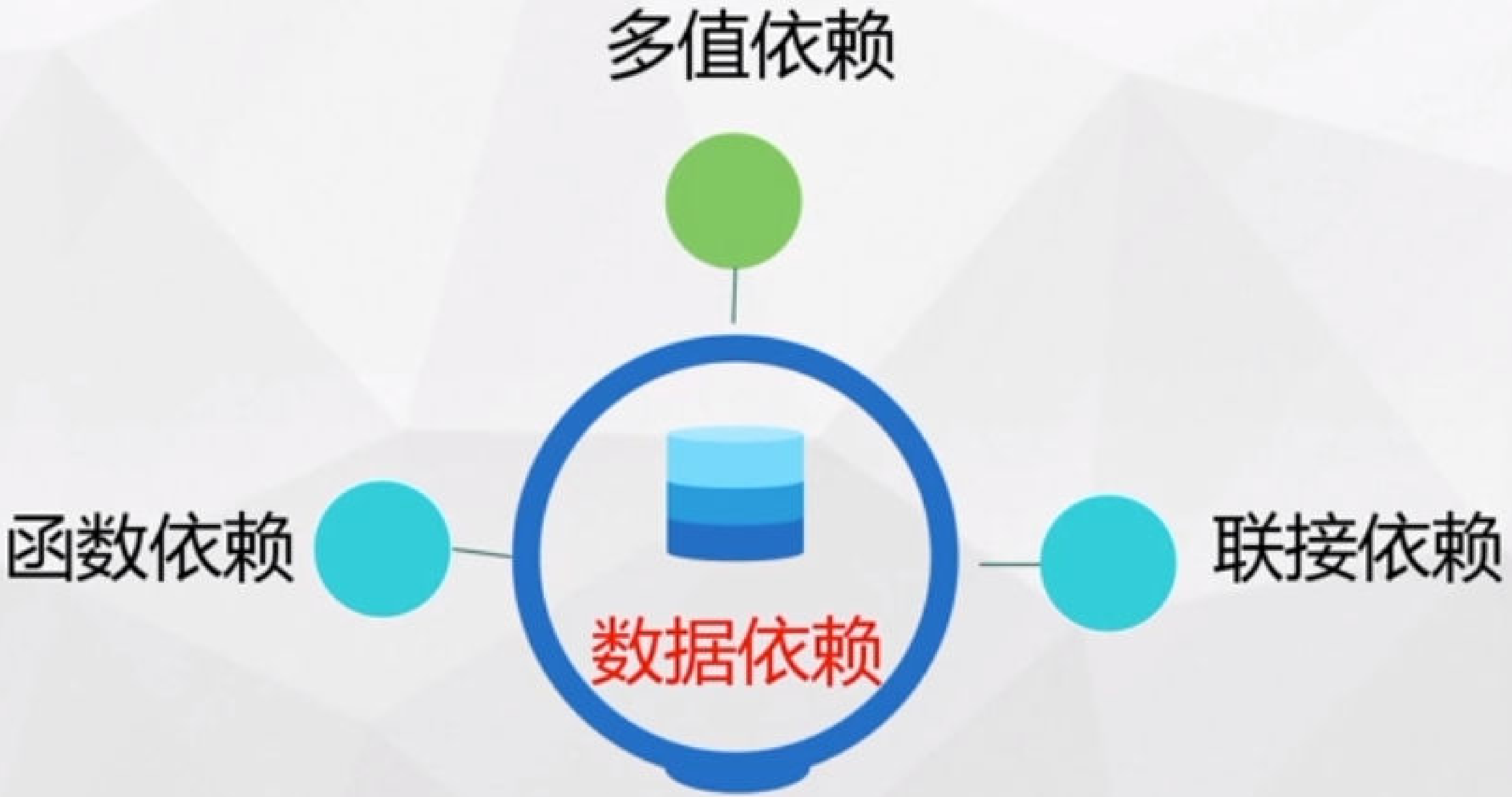
数据依赖

数据依赖是属性-联系数据库设计方法的基础。



具有实用价值、最重要的数据依赖 → 函数依赖

设有关系模式S(A)，A_i和A_j是属性集A的子集，t是S的任一具体关系，对t中任意两个元组r_m和r_n，如果有$r_m[A_i]=r_n[A_i]$则必定 $r_m[A_j]=r_n[A_j]$，那么称函数依赖$A_i \to A_j$在关系模式S(A)上成立。

$A_i \to A_j$读作“A_i函数决定A_j”，或“A_j函数依赖于A_i”。 如果A_j不函数依赖于A_i，则记作A_i不决定A_j，也就是在→上打一个小撇。

设考官表examiner有属性erid, ername, ersex, erage, dname，主键是erid，分别代表考官号、姓名、性别、年龄和所在院系名。

设考官表examiner有属性erid, ername, ersex, erage, dname，主键是erid，分别代表考官号、姓名、性别、年龄和所在院系名。

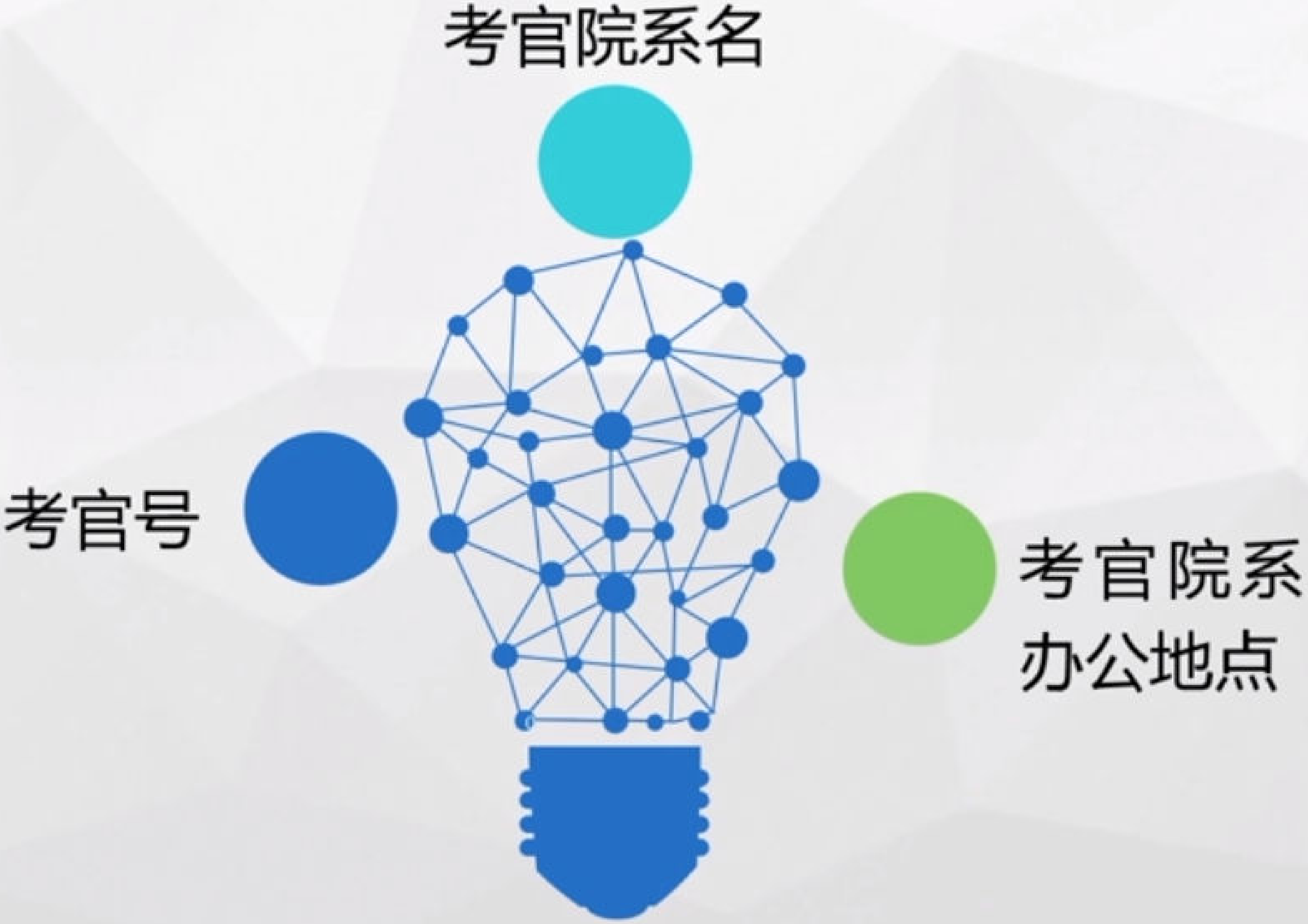
- erid → ername
- erid → ersex
- erid → erage
- erid → dname

由于不同院系可能存在相同姓名的考官，所以不存在函数依赖：

- ername →dname

ername →dname

考官院系



考官院系

考官号→考官院系名

考官院系名→考官院系办公地点

考官名→考官院系办公地点

考官号→考官院系名
考官院系名→考官院系办公地点

推导 →

考官号→考官院系办公地点

逻辑蕴涵

给定关系模式S的函数依赖集D，可以证明其它一些函数依赖也成立，就称这些被证明成立的函数依赖是被D逻辑蕴涵

闭包

给定关系模式S的函数依赖集D， D逻辑蕴涵的所有函数依赖的集合称为D的闭包，记作D^+

Armstrong进行了总结，提出把其中三条作为公理，这就是著名的Armstrong公理。

反射律

若$A_j \subseteq A_i$，则$A_i \to A_j$。

(erid,ername)→ername

(erid,ername)→erid

(erid,ername)→erid

增广律

若$A_i \to A_j$ ，则$A_iA_k \to A_jA_k$。

比如，erid→ername成立，按照增广律，(erid,erage)→(ername,erage)也成立

传递律

若$A_i \to A_j$，$A_j \to A_k$，则$A_i \to A_k$。

那么Ai函数决定Ak

传递律示例

考官院系名→考官院系办公地点

考官号→考官院系名

考官号→考官院系公地点

ARMSTRONG公理

反射律
增广律
传递律

可以证明Armstrong公理是**正确的、完备的**。Armstrong公理的正确性保证推出的所有函数依赖是正确的；完备性保证可以推出所有被蕴涵的函数依赖。换句话说，Armstrong公理能够保证函数依赖推导的**有效性和可靠性**。

有三条非常实用的推论

合并规则
分解规则
伪传递规则

合并规则

若$A_i \rightarrow A_j$， $A_i \rightarrow A_k$，则$A_i \rightarrow A_jA_k$。

证明：

$$\left.\begin{aligned} A_i \rightarrow A_j &\Rightarrow A_i \rightarrow A_iA_j \\ A_i \rightarrow A_k &\Rightarrow A_iA_j \rightarrow A_jA_k \end{aligned}\right\} \Rightarrow A_i \rightarrow A_jA_k$$

分解规则

若$A_i \rightarrow A_jA_k$，则$A_i \rightarrow A_j$， $A_i \rightarrow A_k$ 。

证明:

$$\left.\begin{aligned} A_j \subseteq A_jA_k \Rightarrow\ & A_jA_k \rightarrow A_j \\ & A_i \rightarrow A_jA_k \end{aligned}\right\} \Rightarrow A_i \rightarrow A_j$$

同理： $A_i \rightarrow A_k$

伪传递规则

若$A_i \rightarrow A_j$，$A_j A_l \rightarrow A_k$，则$A_i A_l \rightarrow A_k$。

证明:

$$A_i \rightarrow A_j \Rightarrow \left.\begin{array}{l} A_i A_l \rightarrow A_j A_l \\ A_j A_l \rightarrow A_k \end{array}\right\} \Rightarrow A_i A_l \rightarrow A_k$$

属性闭包

设D为属性集A上的一组函数依赖，α是A的子集，属性集α关于函数依赖集D的闭包写作α， α关于函数依赖集D的闭包就是能由D根据Armstrong公理推导出的左部为α的所有函数依赖的右部组成的集合，也就是能由D根据Armstrong公理推导出α能够函数决定的所有属性的集合。

对$S<A,D>$，假设M是A的子集, M包括属性$\{A_1,A_2,\ldots A_k\}$)
求属性集M，关于A上的函数依赖集D的闭包M_D^+

输入： A，M，D

输出： M关于D的闭包

步骤

(1) 用N保存闭包计算结果，首先将N初始化为M，即N包括属性$\{A_1,A_2,\ldots A_k\}$。

(2) 在D中反复寻找这样的函数依赖：B→C，其左部B是N的子集但C不是N的子集，如果找到了这样的B→C就把右部C并入N中，并重复这个过程。

(3) 当最终再也不能添加任何属性时，集合N就是M关于D的闭包。

对S<A,D>，A={a,b,c,g,h,i},D={a→b,a→c,cg→h,cg→i,b→h}。
计算$(ag)_D^+$

所用依赖	$(ag)_D^+$
初始	ag
a→b	agb
a→c	agbc
cg→h	agbch
cg→i	agbchi

$(ag)_D^+=agbchi$

所以(ag)关于D的闭包=agbchi

如果$G^+ = H^+$

$\Downarrow$

H与G等价

H与G等价就意味着H中的每个函数依赖属于G^+，且G中的每个函数依赖属于H^+。

H与G等价就意味着H中的每个函数依赖属于G+

(1)D中任一函数依赖的右部仅含有一个属性。

(2)D中不存在这样的函数依赖M→N，函数依赖左部M有真子集Q使得D-{M→N}∪{Q→N}与D等价。

(3)D中不存在这样的函数依赖M→N，使得D-{M→N}与D等价。

D中不存在这样的函数依赖M→N

满足这三个条件的函数依赖集我们就称其为

最小函数依赖集或极小函数依赖集

事实上，每一个函数依赖集都有等价的极小函数依赖集D_m，极小函数依赖集的定义正好给出了计算一个给定函数依赖集D的极小函数依赖集方法。

(1)逐一检查D中各函数依赖d_i：$M \rightarrow N$，若$N = A_1A_2 \ldots A_k$，$k \geq 2$，则用｛$M \rightarrow A_j \mid j=1，2，\ldots，k$｝来取代$M \rightarrow N$。

(2)逐一检查D中各函数依赖d_i：$M \rightarrow N$，设$M = B_1B_2 \ldots B_m$，逐一考查B_i （i=l，2，…，m），若$N \in (M - B_i)_D^+$，则以$M-B_i$取代M。

(3) 逐一检查D中各函数依赖d_i：$M \rightarrow N$，令$G = D-\{M \rightarrow N\}$， 若$G^+ = D^+$, 则从D 中去掉此函数依赖。

$$D = \{a \rightarrow b,\ \cancel{b \rightarrow a},\ b \rightarrow c,\ \cancel{a \rightarrow c},\ c \rightarrow a\}$$

$$D_{m1} = \{a \rightarrow b,\ b \rightarrow c,\ c \rightarrow a\}$$

$$D = \{a\to b,\ b\to a,\ \cancel{b\to c},\ a\to c,\ c\to a\}$$

$$D_{m2} = \{a\to b,\ b\to a,\ a\to c,\ c\to a\}$$

启示

一个函数依赖集的极小依赖集不唯一；一个函数依赖集与其极小依赖集以及其闭包都是等价的。

在关系模式S(A)中，对于A的子集A_i和A_j，如果$A_i \rightarrow A_j$，但A_j不是A_i的子集，则称$A_i \rightarrow A_j$是非平凡的函数依赖；如果$A_i \rightarrow A_j$，但A_j是A_i的子集，则称$A_i \rightarrow A_j$是平凡的函数依赖。

比如考官表中，erid→ername是非平凡的函数依赖，erid→erid、(erid, ername)→erid都是平凡的函数依赖。除非特别说明，一般总是只考虑非平凡的函数依赖。

模式分解

关系数据库设计的属性-联系方法，就是把需要数据库保存的所有属性放在一张关系表中，进而基于数据依赖来优化这个模式，得到期望的结果，这一过程的基本操作就是模式分解。

模式分解

中国大学MOOC

设关系模式S<A,D>，属性全集为A，函数依赖集为D。

而$S_1<A_1,D_1>$，$S_2<A_2,D_2>$，…，$S_n<A_n,D_n>$，其中$A=A_1A_2\ldots A_n$，且不存在$A_i \subseteq A_j$,

$D_i=\{\alpha\rightarrow\beta \mid \alpha\rightarrow\beta\in D^+ \wedge \alpha\beta\subseteq A_i\}$ $(i,j=1,2,\ldots\ldots n)$，

对于S的任意关系t, $t_i=\pi_{S_i}(t)$。

关系模式S_1，……，S_n的集合用ρ表示，$\rho=\{S_1，\ldots\ldots，S_n\}$。

用ρ代替S的过程称为关系模式S的模式分解。