关系理论

1、函数依赖

(1) 非平凡的函数依赖：$X \to Y$，$Y \not\subseteq X$

(2) 平凡的函数依赖：$X \to Y$，$Y \subseteq X$

无特殊说明下，均讨论非平凡的函数依赖。即X可以推出Y，但Y不是X的子集。因为一般某个集合总能推出其子集（这种情况就是平凡的函数依赖），没啥用。

(3) 完全函数依赖：$X \to Y$，并且对于X的任意真子集X'，都有$X' \not\to Y$。则称Y完全函数依赖于X。记作$X \xrightarrow{F} Y$。

(4) 部分函数依赖：Y不完全函数依赖于X。记作$X \xrightarrow{P} Y$。例如$A \to C$，又有$AB \to C$，那么C就是部分函数依赖于AB的，这种情况会造成数据冗余。

2、码

(1) 候选码：是一个属性组（或者属性），通过该属性组能推出所有的属性，并且该属性组的任意子集都不能再推出所有属性了。即在满足完全函数依赖的前提下，还得是最小的属性组。

求所有候选码的方法：

例：集合$U=\{A, B, C, D, E, G\}$，函数依赖集$F=\{AB \to C, CD \to E, C \to A, A \to G\}$

【Step 1】

找出一定属于候选码的属性，可能属于候选码的属性，以及不属于候选码的属性。方法如下：

一定属于候选码的属性：只出现在左边，或者左右都没出现

可能属于候选码的属性：左右都出现

不属于候选码的属性：只出现在右边

【例题分析】

只出现在左边的是B和D，没有左右都没出现的属性，所以BD一定是属于候选码的属性。

左右都出现的有A，C，E，因此这三个是可能属于候选码的属性，即待定的备选。

只出现在右边的有G，因此G是不属于候选码的属性，可以不管了。

【Step 2】

先对确定的属性求闭包，若不能构成候选码，再将确定的属性和待定的属性进行组合，做闭包运算，直到得到的属性组能够推出全部的属性。

闭包运算：

若要求某属性组的闭包，首先设有集合X，令X={该属性组}。

$X^{(0)}$=自身

$X^{(1)}$=$X^{(0)}$中的属性所能推出的

当$X^{(1)}$不等于$X^{(0)}$时，$X^{(2)}$=$X^{(1)}$中的属性能所推出的

依次类推…直到$X^{(n)}$=U或者$X^{(n)}$=$X^{(n-1)}$就求得了属性组的闭包$(X)_F^+$。

ps.闭包运算还可用于判断X→Y是否成立：当$Y \subseteq (X)_F^+$时，有X→Y。

【例题分析】

根据step1的分析，一定是候选码的为BD。可能是候选码的有A、C、E。

于是先对BD求闭包（这里可求得BD推不出全部的属性），因此再分别对BDA，BDC，BDE进行闭包运算，看其是否能得到全部属性。如若不能，再增加如BDAC，BDAE之类的组合，直到求出候选码为止。

以BDA为例：设X={BDA}

$X^{(0)}$=BDA

$X^{(1)}$=BDACG

∵$X^{(1)} \neq X^{(0)}$，有$X^{(2)}$=BDACGE

因此$(BDA)_F^+$为U，所以$(BDA)_F^+$是候选码

全部进行完闭包运算后，可知集合U在F下的候选码为{(BDA),(BDC),(BDE)}

(2) 超码：能推出所有属性的属性组的集合，根据概念可知，候选码是极小的超码集，是超码的子集

(3) 主码：当有多个候选码时（如例题那样），挑出一个作为主码，简称码

(4) 主属性：包含在任何一个候选码中的属性，如例题中ABCDE都是主属性

(5) 非主属性：不包含在任何一个候选码中的属性，如例题中的G

(6) 外码：关系模式R中，若有一个属性或属性组X，它不是R的码，但X是另一个关系模式S中的码，称X是R的外码

(7) 全码：最极端情况下，整个属性组都是码，称为全码

3、范式

（1）1NF：所有属性都是不可分割的数据项

如果某个属性，例如学校，还可以继续拆分为高中和大学，就不满足1NF了。1NF是关系数据库需要满足的最低要求

（2）2NF：在满足1NF的前提下，不包含非主属性对码的部分函数依赖（即每一个非主属性都完全函数依赖于码）

例如在关系R中，码是学号和班级，非主属性是姓名，因为通过学号就能直接推出姓名了，不需要班级，此处姓名就部分依赖于码了，不满足2NF

（3）3NF：在满足2NF的前提下，不包含非主属性对码的传递函数依赖（即码应该直接决定非主属性，不能间接决定）

传递函数依赖：若$X \to Y$，$Y \to Z$，且$Z \neq Y$，$Y \not\to X$，有$X \to Z$，此时称Z对X有传递函数依赖。

例如在关系R中，码是客户姓名，非主属性是订单编号和订单负责人，通过客户姓名可以推出他的订单编号，再通过订单编号能推出订单负责人，这种情况下客户姓名和订单负责人是间接决定的，存在传递函数依赖，不满足3NF

（4）BCNF：消除任何属性对候选码的传递依赖，即每一个决定因素都包含码，表现为在函数依赖集当中，左边的都包含候选码（整个属性组！）

（5）4NF（应该不考这个）：不允许有非平凡且非函数依赖的多值依赖

多值依赖（个人理解，仅供参考，我觉得不会细考）：X，Y，Z属于集合U，且$Z=U-X-Y$。当给定一组（x，z）值的时候，可以确定一组Y的值，但这组Y的值仅仅取决于x，此时有$X \to\to Y$。其实这里就是存在了一对多的关系，即一个x和一组z有关，但x并不能唯一确定一个z，通过x和z能找到一组y，但你只通过x也能确定y。

平凡的多值依赖：Z是空集

非平凡的多值依赖：Z不是空集

判断方法与分解方法：

R为{A，B，C，D}

2NF（没有部分函数依赖）：若码是AB，F中若为{A→C，AB→D}，对于C，只需要A就能推出，那么C部分函数依赖于码AB，这种情况就不是2NF。

若要分解为2NF，只需将不符合要求的拿出来，即分为R_1{A，B，D}和R_2{A，C}

3NF（没有部分函数依赖与传递函数依赖）：若码是AB，F若为{AB→C，C→D}，这里不存在部分函数依赖。但是对于D，需要AB推出C后才能间接推出D，那么D传递函数依赖于AB，不满足3NF。

若要分解为3NF，同样将不符合要求的拿出来，即分为R1{A，B，C}和R2{C，D}。

BCNF（没有部分函数依赖，同时每一个决定因素都包含码）：

若R是（A，B，C），F是{AC→B，AB→C，B→C}，候选码则是AC和AB。这里不存在部分函数依赖，但对于B→C来说，决定因素B不包含码，因此它不是BCNF。

4、最小函数依赖集

求最小函数依赖集的方法

step 1：拆分右侧

例如将A→BC拆为A→B和A→C

step 2：去除自身求闭包

若有有AB→C，BC→E，AE→G，去除AB自身能推出的C，基于剩余的依赖关系求AB的闭包，若AB通过剩余的关系也能求出C，那么删除AB→C这个依赖关系

step 3：左侧最小化

例如目前保留的关系有ABC→D，观察左边的ABC当中，A是否能由BC推出，B是否能由AC推出，C是否能由AB推出。假设C能被AB推出，那么左侧去掉C，更新为AB→D。

例：设F={C→A，CG→BD，CE→A，ACD→B}，求最小函数依赖集。

step 1：

将CG→BD拆分为CG→B和CG→D。

step 2：

$(C)_F^+=C$，因此保留C→A。$(CG)_F^+=CGADB$，因此去掉CG→B。$(CG)_F^+=CGA$，因此保留CG→D。$(CE)_F^+=CEA$，因此去掉CE→A。$(ACD)_F^+=ACD$，因此保留ACD→B。

step 3：

C→A已经是最小。CG→D已经是最小。ACD当中，C可以推出A，去掉A，更新为CD→B。

因此，本题的最小函数依赖集为{C→A，CG→D，CD→B}。

5、模式分解

(1) 模式分解的准则：无损连接、保持函数依赖

(2) 无损连接：分解后再次自然连接，与分解前相同

判断无损连接的方法

step 1：画表格。列表示所有的属性，有多少属性就画多少个属性列。行表示分解后的关系，有几个关系就画几个关系行。

step 2：根据每一行关系进行判断，找到关系中的每个属性对应第几列，并在相应的位置上标为a_j，下标j是表格里的列数。其余关系中不存在的属性则标为b_{ij}，ij是表格对应的行数和列数。

step 3：依次对函数依赖集里的各个依赖关系进行考察。例如有XY→Z。在属性列中找到X和Y，观察X和Y的行列上是否有相同的标记（b的下标要相同）。若有，则查看它们对应在属性列Z上的各个标记。其中若有a_j，则将属性列上的这些标记全部改为a_j。若没有a_j，则找到值最小的b_{ij}，将这些标记全部改为b_{ij}。

step 4：反复执行以上操作，直到某一行全部变为a为止，则表明具有无损连接性。否则不具有无损连接性。

例：F={A→C，C→D，B→C，DE→C，CE→A}。分解为R_1(AD)，R_2(AB)，R_3(BC)，R_4(CDE)，R_5(AE)。

step 1：画表格

	A	B	C	D	E
R_1	a_1	b_{12}	b_{13}	a_4	b_{15}
R_2	a_1	a_2	b_{23}	b_{24}	b_{25}
R_3	b_{31}	a_2	a_3	b_{34}	b_{35}
R_4	b_{41}	b_{42}	a_3	a_4	a_5
R_5	a_1	b_{52}	b_{53}	b_{54}	a_5

A→C

step 3：更新表格

	A	B	C	D	E
R_1	a_1	b_{12}	b_{13}	a_4	b_{15}
R_2	a_1	a_2	b_{13}	b_{24}	b_{25}
R_3	b_{31}	a_2	a_3	b_{34}	b_{35}
R_4	b_{41}	b_{42}	a_3	a_4	a_5
R_5	a_1	b_{52}	b_{13}	b_{54}	a_5

C→D

step 4：反复更新表格

	A	B	C	D	E
R_1	a_1	b_{12}	b_{13}	a_4	b_{15}
R_2	a_1	a_2	b_{13}	a_4	b_{25}
R_3	b_{31}	a_2	a_3	a_4	b_{35}
R_4	b_{41}	b_{42}	a_3	a_4	a_5
R_5	a_1	b_{52}	b_{13}	a_4	a_5

B→C

step 4：反复更新表格

	A	B	C	D	E
R_1	a_1	b_{12}	b_{13}	a_4	b_{15}
R_2	a_1	a_2	a_3	a_4	b_{25}
R_3	b_{31}	a_2	a_3	a_4	b_{35}
R_4	b_{41}	b_{42}	a_3	a_4	a_5
R_5	a_1	b_{52}	b_{13}	a_4	a_5

DE→C

step 4：反复更新表格

	A	B	C	D	E
R_1	a_1	b_{12}	b_{13}	a_4	b_{15}
R_2	a_1	a_2	a_3	a_4	b_{25}
R_3	b_{31}	a_2	a_3	a_4	b_{35}
R_4	b_{41}	b_{42}	a_3	a_4	a_5
R_5	a_1	b_{52}	a_3	a_4	a_5

CE→A

step 4：反复更新表格

	A	B	C	D	E
R_1	a_1	b_{12}	b_{13}	a_4	b_{15}
R_2	a_1	a_2	a_3	a_4	b_{25}
R_3	b_{31}	a_2	a_3	a_4	b_{35}
R_4	a_1	b_{42}	a_3	a_4	a_5
R_5	a_1	b_{52}	a_3	a_4	a_5

二轮扫描 →

step 4: 仅复更新表格

	A	B	C	D	E
R_1	a_1	b_{12}	a_3	a_4	b_{15}
R_2	a_1	a_2	a_3	a_4	b_{25}
R_3	b_{31}	a_2	a_3	a_4	b_{35}
R_4	a_1	b_{42}	a_3	a_4	a_5
R_5	a_1	b_{52}	a_3	a_4	a_5

A→C

step 4: 仅复更新表格

	A	B	C	D	E
R_1	a_1	b_{12}	a_3	a_4	b_{15}
R_2	a_1	a_2	a_3	a_4	b_{25}
R_3	b_{31}	a_2	a_3	a_4	b_{35}
R_4	a_1	b_{42}	a_3	a_4	a_5
R_5	a_1	b_{52}	a_3	a_4	a_5

C→D 没有更新

step 4: 仅复更新表格

	A	B	C	D	E
R_1	a_1	b_{12}	a_3	a_4	b_{15}
R_2	a_1	a_2	a_3	a_4	b_{25}
R_3	b_{31}	a_2	a_3	a_4	b_{35}
R_4	a_1	b_{42}	a_3	a_4	a_5
R_5	a_1	b_{52}	a_3	a_4	a_5

B→C 没有更新

step 4: 仅复更新表格

	A	B	C	D	E
R_1	a_1	b_{12}	a_3	a_4	b_{15}
R_2	a_1	a_2	a_3	a_4	b_{25}
R_3	b_{31}	a_2	a_3	a_4	b_{35}
R_4	a_1	b_{42}	a_3	a_4	a_5
R_5	a_1	b_{52}	a_3	a_4	a_5

---

step 4: 仅复更新表格

	A	B	C	D	E
R_1	a_1	b_{12}	a_3	a_4	b_{15}
R_2	a_1	a_2	a_3	a_4	b_{25}
R_3	b_{31}	a_2	a_3	a_4	b_{35}
R_4	a_1	b_{42}	a_3	a_4	a_5
R_5	a_1	b_{52}	a_3	a_4	a_5

DE→C 没有更新

step 4: 仅复更新表格

	A	B	C	D	E
R_1	a_1	b_{12}	a_3	a_4	b_{15}
R_2	a_1	a_2	a_3	a_4	b_{25}
R_3	b_{31}	a_2	a_3	a_4	b_{35}
R_4	a_1	b_{42}	a_3	a_4	a_5
R_5	a_1	b_{52}	a_3	a_4	a_5

CE→A 没有更新

step 4: 仅复更新表格

	A	B	C	D	E
R_1	a_1	b_{12}	a_3	a_4	b_{15}
R_2	a_1	a_2	a_3	a_4	b_{25}
R_3	b_{31}	a_2	a_3	a_4	b_{35}
R_4	a_1	b_{42}	a_3	a_4	a_5
R_5	a_1	b_{52}	a_3	a_4	a_5

三轮扫描 没有更新

step 4: 仅复更新表格

	A	B	C	D	E
R_1	a_1	b_{12}	a_3	a_4	b_{15}
R_2	a_1	a_2	a_3	a_4	b_{25}
R_3	b_{31}	a_2	a_3	a_4	b_{35}
R_4	a_1	b_{42}	a_3	a_4	a_5
R_5	a_1	b_{52}	a_3	a_4	a_5

循环终止，没有出现全为a的行，表明该分解不具有无损连接性

分解数据库为3NF，并保持无损连接和函数依赖的方法（不一定考）

设有属性集U，函数依赖集F，其模式分解后应得到多个U的子集

step 1: 求出最小函数依赖集F_{min}

step 2: 观察U，找出其中未在F_{min}中出现过的属性，将其分为一个集合。

例如U中有ABCDEG，但F_{min}中没有关于G的依赖关系，则划分出{G}

step 3: 观察F_{min}，若有多个依赖关系的决定因素（即左侧）相同，则均划分到同一个集合中。若没有相同的，则仅将该依赖关系划分到同一个集合。

例如F_{min}中有A→B，A→C，D→E，则划分出{ABC}和{DE}

step 4: 求出F_{min}的候选码，若候选码未上述分类中出现，则单独将候选码分为一类。

例如上述F_{min}的候选码为ADG，可知其未出现在各分类中，因此再划分一个集合{ADG}。

由上述举例可知，其最终的模式分解为{G} {ABC} {DE} {ADG}

3.关系

一般来说，在关系模型中D1，D2，Dn的笛卡尔积是没有实际语义的，只有它的某个真子集才有实际意义

关系有三种类型：
基本关系（基本表或基表）
、查询结果和视图

基本关系有6条性质：
①列是同质的（homogeneous），即每一列中的分量是同一类型的数据，来自同一个域。
②不同的列可出自同一个域，称其中的每一列为一个属性，不同的属性要给予不同的属性名。例如，在上面的例子中，也可以只给出两个域：
人（PERSON）={张清玫，刘逸，李勇，刘晨，王敏}
专业（MAJOR）={计算机科学与技术，信息管理与信息系统}
SMP关系的导师属性和研究生属性都是从PERSON 域中取值。为了避免混淆，必须给这两个属性取不同的属性名，而不能直接使用域名。因此，定义导师属性名为SUPERVISOR，研究生属性名为POSTGRADUATE。
③列的顺序无所谓，即列的次序可以任意交换。由于列顺序是无关紧要的，因此在许多实际的关系数据库产品中增加新属性时，永远是将其插至最后一列。
④任意两个元组的码不能取相同的值。
⑤行的顺序无所谓，即行的次序可以任意交换。
⑥分量必须取原子值，即每一个分量都必须是不可分的数据项。

关系操作的特点是集合操作方式，即操作的对象和结果都是集合

关系模型有三类完整性约束：实体完整性，参照完整性，用户定义完整性

关系的两个不变性

关系语言

1. 关系代数语言

1.域
域是一组具有相同数据类型的值的集合

(1) 集合运算符（设有关系R和关系S）

并∪：R并S，即由属于R或S的元组构成，同时去掉重复的元组

差−：R差S，即由属于R但不属于S的元组构成

交∩：R交S，即由既属于R又属于S的元组构成

2. 笛卡尔积×：即由R中的每个元组与S中的所有元组进行组合

(2) 关系运算符

选择σ：得到表中的指定行，写作$\sigma_{条件}$(表名)

投影π：得到表中的指定列，写作$\pi_{列名}$(表名)，投影后要去除重复行

连接⋈：将两个表根据指定条件连接在一起，写作$R \underset{A\theta B}{\bowtie} S$

等值连接是指条件为属性R.A=S.B

自然连接是指条件为属性R.A=S.A，并且要去掉重复列，写作R ⋈ S

悬浮元组是指自然连接时由于S中不匹配而在R中被舍弃的元组

外连接是指保留悬浮元组的连接，不匹配的位置填NULL，写作⟗

左外连接是指只保留R中悬浮元组的连接，写作⟕

右外连接是指只保留S中悬浮元组的连接，写作⟖

不考

除÷：设R和S除运算的结果为T，则T包含所有在R中但不在S中的属性和值，且T的元组与S的元组经过组合均能出现在R中

例：

R			S			R÷S
A	B	C	B	C	D	A
a_1	b_1	c_2	b_1	c_2	d_1	a_1
a_2	b_3	c_7	b_2	c_1	d_1	
a_3	b_4	c_6	b_2	c_3	d_2	
a_1	b_2	c_3				
a_4	b_6	c_6				
a_2	b_2	c_3				
a_1	b_2	c_1				

a_2中虽然也出现了S中的b_2 c_3，但是a_2与S中其余的b_1 c_2和b_2 c_1的组合并没有出现在R中

关系代数解题方法

（1）常规题（求某几个属性特定值）

格式一般为π__ __（σ__ __（表名⋈表名））

投影运算的下标为题目要求的最终需要的列，选择运算的下标为题目给出的属性条件。并行的条件之间用逗号隔开，条件表达式的运算符有¬ ∧ ∨ <>（不等于）≥ ≤ > < =。若有多个表，表之间常用自然连接。

（¬非 ∧与 ∨或）

（2）除运算（求满足某属性全部值的其他属性）

这种题是指求是满足B表某属性全部值的在A表上的其他属性。这是除运算的特性，因此在出现"全部"二字时，需要用除运算完成。通常分别对A和B做投影运算，再对生成的方表进行除运算。

A中包含属性x和y，B中包含属性y，且B中属性y的值为全集且无重复，求全部y的x写作：$\pi_{x,y}(A) \div \pi_{y}(B)$

例如A表为学生选课表（属性包括学号和所选的课程），B表为课程信息表（属性包括课程），求选了全部课程的学生学号。全部课程只在B表出现，学号只在A表出现。于是先全选A表的学号字段和课程字段，再全选B表的课程字段，将二者相除：$\pi_{学号,课程}(A) \div \pi_{课程}(B)$

（3）差运算

例：有学生表SC，包含属性姓名、成绩，求没有任何一门课程低于80分的学生的姓名。

思路：可以先求有课程低于80分的学生姓名，再用全表相减。

$\pi_{姓名}(SC) - \pi_{姓名}(\sigma_{成绩<80}(SC))$

2、元组关系演算语言

（1）元组演算表达式：$\{t \mid \phi(t)\}$，其中t是元组变量，$\phi(t)$是公式，它由原子公式和运算符组成。

（2）原子公式

1、$R(t)$ 表示t是关系R中的元组

2、$t[i]\theta u[j]$ 表示元组t的第i个分量和元组u的第j个分量满足比较关系θ

3、$t[i]\theta c$ 或 $c\theta t[i]$ 表示元组t的第i个分量和常量c满足比较关系θ

（3）运算符（按优先级从高到低书写）

1、比较运算符：> = <

2、量词运算符：包括∃和∀。其中∃的优先级大于∀

3、逻辑运算符：包括¬和∧和∨。其中¬的优先级大于∧，∧的优先级大于∨

关系代数语言和元组演算语言的转换：

（1）并

$R\cup S=\{t\mid R(t)\vee S(t)\}$

（2）交

$R\cap S=\{t\mid R(t)\wedge S(t)\}$

（3）差

$R-S=\{t\mid R(t)\wedge\neg S(t)\}$

（4）笛卡尔积

$R\times S=\{t^{(n+m)}\mid (\exists u^{(n)})(\exists v^{(m)})R(u)\wedge S(v)\wedge t[1]=u[1]\wedge\cdots\cdots\wedge t[n]=u[n]\wedge t[n+1]=v[1]\ \cdots\cdots\wedge t[n+m]=\cdots[m]\}$

其中R有n个属性，S有m个属性，根据笛卡尔积的定义，t的目数为n+m（即有n+m个属性）

（5）投影

$\pi_{i1,i2,\cdots,ik}(R)=\{t^{(k)}\mid (\exists u)(R(u)\wedge t[1]=u[i1]\wedge\cdots\cdots\wedge t[k]=u[ik]\}$

表示最终需要k列，因此t的目数为k。选取中间变量u，令u为R中的全部元组，令结果集t的第一列为R中需要的第一列（即i1），最后一列k列为R中的ik列

（6）选择

$\sigma_F(R)=\{t\mid R(t)\wedge F'\}$

F是选择条件，F'是F等价代换后的元组演算表达式

例题：

1、查询Student表中IS系的全体学生，其中学生所在系为第五列属性。

$S_{IS}=\{t\mid Student(t)\wedge t[5]='IS'\}$

上式表示设结果集为S_{IS}，其中的元组t满足条件：t属于Student表且第五列为IS。

2、查询Student表中学生的姓名和所在系，其中姓名为第二列，所在系为第五列。

$S_1 = \{t^{(2)} \mid (\exists u)(Student(u) \wedge t[1]=u[2] \wedge t[2]=u[5])\}$

上式表示设结果集为S_1，其中的元组t有两列属性，这两列属性满足条件：设有元组u，u是Student表中的元组，结果集的第一列（即t[1]）为Stusent表的第二列（即u[2]），结果集的第二列为Student表的第五列。

解题格式：

首先设结果集（例如设为S），令其中的元组为t。若题目中指明了需要哪些属性时，需要标注t的目数。当需要用量词运算符时，记得前后用括号括起来。各条件之间一般用交运算。在元组表达式中，记得首先要指出所设元组属于哪个关系。

$S = \{t \mid (量词运算)(指出元组所属的关系 \wedge 元组需要满足的条件)\}$

补：把不产生无限关系的表达式称为安全表达式，所采取的措施称为安全措施

事务调度

1、事务调度的准则

（1）一组事务的调度必须保证：

包含了所有事务的操作指令；一个事务内部的指令顺序必须保持不变

（2）并行事务调度必须保证：

可串性化，将所有可能的串行调度结果推演一遍，对于某个具体的并行调度再执行一遍，看是否能与某个串行调度的结果相同

（3）判断可串性化的充分条件：冲突可串性化（冲突可串性化一定是可串性化调度，但可串性化调度不一定是冲突可串性化）

冲突操作：不同事务对同一数据分别进行读和写；不同事务对同一数据分别进行写和写

冲突可串性化调度即不交换不同事务的冲突操作次序，也不交换同一事务的两个操作的次序。但可以交换不同事务对不同数据各种操作次序，也可以交换不同事务对同一数据的读取操作次序

2、封锁

（1）X锁：写锁，某事务对数据对象上锁后，可读取和修改该数据对象，其他事务不可再对该数据对象添加D锁

表示方法：上锁Xlock（）释放锁Unlock（）

（2）S锁：读锁，某事务对数据对象上锁后，可读取但不可修改该数据对象，其他事务可以对该数据对象添加DS锁，但不能添加DX锁

表示方法：上锁Slock（）释放锁Unlock（）

（3）封锁协议

一级封锁协议：写前加D写锁，事务结束释放写锁；可防止丢失修改

二级封锁协议：写前加D写锁，读前加D读锁，读完释放读锁，事务结束释放写锁；可防止丢失修改和读脏数据

三级封锁协议（常用：支持一致性维护）：写前加D写锁，读前加D读锁，事务结束释放各锁；可防止丢失修改、读脏数据和不可重复读

如果所有事务均遵循三级封锁协议，由于其隔离级别高，那么这些事务无论怎样交叉并行，都是可串性化的调度

(4) 两段锁协议 (2PL)

三级封锁协议可以保证并发操作的正确性，但由于其太过严苛，对并发度有负面影响。三级封锁协议实际是两段锁协议的特例，是更严格的两段锁协议

两段锁协议要求：事务在对任何数据进行读写前，需要获得对该数据的封锁；而当事务在释放任何一个封锁后，不可再获得任何其他封锁

事务遵循两段锁协议是可串性化的充分条件，遵循两段锁协议是可能发生死锁的

数据库设计

1、画E-R图（概念结构设计）

实体： 

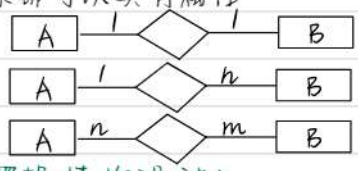
关系： 

属性：  注意：实体和关系都可以具有属性

1对1联系：1个A对应1个B

1对n联系：1个A对应n个B

n对m联系：n个A对应m个B



2、E-R图转换为关系模型（逻辑结构设计）

第一步：将各个实体的名字转换为各个关系模式的名字

第二步：实体的属性就是关系的属性，实体的码就是关系的码

第三步：实体间联系的转换

1对1联系：在任意一方加入对方的主码并设为其外码，并加入联系本身的属性

1对n联系：将1方的主码加入n方作为外码，并同时将联系的属性加入n方

n对m联系：将联系本身转换为一个关系模式，将联系双方的主码加入其中设为码，并将联系的属性也加入其中