

# 关系理论

## 1. 函数依赖

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

(2) 平凡的函数依赖:  $X \rightarrow Y, Y \subseteq X$

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。因为一般某个集合尚能推出其子集(这种情况就是平凡的函数依赖), 没啥用。

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

(4) 部分函数依赖:  $Y$ 不完全函数依赖于 $X$ 。记作 $X \twoheadrightarrow Y$ 。例如 $A \rightarrow C$ , 又有 $AB \rightarrow C$ , 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 $F = \{AB \rightarrow C, CD \rightarrow E, E \rightarrow A, A \rightarrow G\}$

【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 $X = \{ \text{该属性组} \}$ 。

$X^{(0)} = \text{自身}$

$X^{(1)} = X^{(0)}$ 中的属性所能推出的

当 $X^{(1)} \neq X^{(0)}$ 时， $X^{(2)} = X^{(1)}$ 中的属性所能推出的

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

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

### 【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 $X = \{BDA\}$

$X^{(0)} = BDA$

$X^{(1)} = BDACGE$

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

因此  $(BDA)^+ = U$ ，所以  $(BDA)^+$  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 $X \rightarrow Y$ ,  $Y \rightarrow Z$ , 且 $Z \not\rightarrow Y$ ,  $Y \not\rightarrow X$ , 有 $X \rightarrow Z$ , 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 $X \twoheadrightarrow Y$ 。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 $\{A \rightarrow C, AB \rightarrow D\}$ ，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<sub>1</sub>{A, B, D}和R<sub>2</sub>{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 $\{AB \rightarrow C, C \rightarrow D\}$ , 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 $R_1(A, B, C)$ 和 $R_2(C, D)$ 。

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 $\{AC \rightarrow B, AB \rightarrow C, B \rightarrow C\}$ , 候选码则是AC和AB。这里不存在部分函数依赖, 但对于 $B \rightarrow C$ 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 $A \rightarrow BC$ 拆为 $A \rightarrow B$ 和 $A \rightarrow C$

step 2: 去除自身求闭包

若有 $AB \rightarrow C, BC \rightarrow E, AE \rightarrow G$ , 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 $AB \rightarrow C$ 这个依赖关系

step 3: 左侧最小化

例如目前保留的关系有 $ABC \rightarrow D$ , 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 $AB \rightarrow D$ 。

例: 设 $F = \{C \rightarrow A, CG \rightarrow BD, CE \rightarrow A, ACD \rightarrow B\}$ , 求最小函数依赖集。

step 1:

将 $CG \rightarrow BD$ 拆分为 $CG \rightarrow B$ 和 $CG \rightarrow D$ 。

step 2:

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

step 3:

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

因此, 本题的最小函数依赖集为 $\{C \rightarrow A, CG \rightarrow D, CD \rightarrow B\}$ 。

## 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  $a_{ij}$ ，下标  $j$  是表格里的列数。其余关系中不存在的属性则标为  $b_{ij}$ ， $i$  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  $XY \rightarrow Z$ 。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  $i$  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:  $F = \{A \rightarrow C, C \rightarrow D, B \rightarrow C, DE \rightarrow C, CE \rightarrow A\}$ 。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
$R_1$	$a_1$	$b_{12}$	$b_{13}$	$a_4$	$b_{15}$
$R_2$	$a_1$	$a_2$	$b_{23}$	$b_{24}$	$b_{25}$
$R_3$	$b_{31}$	$a_2$	$a_3$	$b_{34}$	$b_{35}$
$R_4$	$b_{41}$	$b_{42}$	$a_3$	$a_4$	$a_5$
$R_5$	$a_1$	$b_{52}$	$b_{53}$	$b_{54}$	$a_5$

step 3: 更新表格

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

step 4: 反复更新表格

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

step 4: 反复更新表格

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

step 4: 反复更新表格

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

step 4: 反复更新表格

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

$B \rightarrow C$

$DE \rightarrow C$

$CE \rightarrow A$

## 二轮扫描

step 4: 反复更新表格

A B C D E

R<sub>1</sub> (a<sub>1</sub>) b<sub>12</sub> a<sub>3</sub> a<sub>4</sub> b<sub>15</sub>

A→C R<sub>2</sub> (a<sub>1</sub>) a<sub>2</sub> a<sub>3</sub> a<sub>4</sub> b<sub>15</sub> C→D

R<sub>3</sub> b<sub>31</sub> a<sub>2</sub> a<sub>3</sub> a<sub>4</sub> b<sub>15</sub> 没有

R<sub>4</sub> (a<sub>1</sub>) b<sub>12</sub> a<sub>3</sub> a<sub>4</sub> a<sub>5</sub> 更新

R<sub>5</sub> (a<sub>1</sub>) b<sub>12</sub> a<sub>3</sub> a<sub>4</sub> a<sub>5</sub>

step 4: 反复更新表格

A B C D E

R<sub>1</sub> a<sub>1</sub> b<sub>12</sub> a<sub>3</sub> a<sub>4</sub> b<sub>15</sub>

R<sub>2</sub> a<sub>1</sub> a<sub>2</sub> a<sub>3</sub> a<sub>4</sub> b<sub>15</sub> B→C

R<sub>3</sub> b<sub>31</sub> a<sub>2</sub> a<sub>3</sub> a<sub>4</sub> b<sub>15</sub> 没有

R<sub>4</sub> a<sub>1</sub> b<sub>12</sub> a<sub>3</sub> a<sub>4</sub> a<sub>5</sub> 更新

R<sub>5</sub> a<sub>1</sub> b<sub>12</sub> a<sub>3</sub> a<sub>4</sub> a<sub>5</sub>

step 4: 反复更新表格

A B C D E

R<sub>1</sub> a<sub>1</sub> b<sub>12</sub> a<sub>3</sub> a<sub>4</sub> b<sub>15</sub>

R<sub>2</sub> a<sub>1</sub> a<sub>2</sub> a<sub>3</sub> a<sub>4</sub> b<sub>15</sub>

R<sub>3</sub> b<sub>31</sub> a<sub>2</sub> a<sub>3</sub> a<sub>4</sub> b<sub>15</sub>

R<sub>4</sub> a<sub>1</sub> b<sub>12</sub> a<sub>3</sub> a<sub>4</sub> a<sub>5</sub>

R<sub>5</sub> a<sub>1</sub> b<sub>12</sub> a<sub>3</sub> a<sub>4</sub> a<sub>5</sub>

step 4: 反复更新表格

A B C D E

R<sub>1</sub> a<sub>1</sub> b<sub>12</sub> a<sub>3</sub> a<sub>4</sub> b<sub>15</sub>

DE→C R<sub>2</sub> a<sub>1</sub> a<sub>2</sub> a<sub>3</sub> a<sub>4</sub> b<sub>15</sub> CE→A

没有 R<sub>3</sub> b<sub>31</sub> a<sub>2</sub> a<sub>3</sub> a<sub>4</sub> b<sub>15</sub> 没有

更新 R<sub>4</sub> a<sub>1</sub> b<sub>12</sub> a<sub>3</sub> a<sub>4</sub> a<sub>5</sub> 更新

R<sub>5</sub> a<sub>1</sub> b<sub>12</sub> a<sub>3</sub> a<sub>4</sub> a<sub>5</sub>

step 4: 反复更新表格

A B C D E

R<sub>1</sub> a<sub>1</sub> b<sub>12</sub> a<sub>3</sub> a<sub>4</sub> b<sub>15</sub>

R<sub>2</sub> a<sub>1</sub> a<sub>2</sub> a<sub>3</sub> a<sub>4</sub> b<sub>15</sub>

R<sub>3</sub> b<sub>31</sub> a<sub>2</sub> a<sub>3</sub> a<sub>4</sub> b<sub>15</sub>

R<sub>4</sub> a<sub>1</sub> b<sub>12</sub> a<sub>3</sub> a<sub>4</sub> a<sub>5</sub>

R<sub>5</sub> a<sub>1</sub> b<sub>12</sub> a<sub>3</sub> a<sub>4</sub> a<sub>5</sub>

step 4: 反复更新表格

A B C D E

三轮 R<sub>1</sub> a<sub>1</sub> b<sub>12</sub> a<sub>3</sub> a<sub>4</sub> b<sub>15</sub>

扫描 R<sub>2</sub> a<sub>1</sub> a<sub>2</sub> a<sub>3</sub> a<sub>4</sub> b<sub>15</sub>

没有 R<sub>3</sub> b<sub>31</sub> a<sub>2</sub> a<sub>3</sub> a<sub>4</sub> b<sub>15</sub>

更新 R<sub>4</sub> a<sub>1</sub> b<sub>12</sub> a<sub>3</sub> a<sub>4</sub> a<sub>5</sub>

R<sub>5</sub> a<sub>1</sub> b<sub>12</sub> a<sub>3</sub> a<sub>4</sub> a<sub>5</sub>

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<sub>min</sub>

step 2: 观察U，找出其中未在F<sub>min</sub>中出现过的属性，将其分为一个集合。

例如U中有ABCDEFG，但F<sub>min</sub>中没有关于G的依赖关系，则划分出{G}

step 3: 观察F<sub>min</sub>，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<sub>min</sub>中有A→B，A→C，D→E，则划分出{ABC}和{DE}

step 4: 求出F<sub>min</sub>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<sub>min</sub>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}

# 关系语言

## 1. 关系代数语言

(1) 集合运算符 (设有关系R和关系S)

并 $\cup$ : 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 $\cap$ : R交S, 即由既属于R又属于S的元组构成

笛卡尔积 $\times$ : 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 $\sigma$ : 得到表中的指定行, 写作 $\sigma_{\text{条件}}$  (表名)

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

连接 $\bowtie$ : 将两个表根据指定条件连接在一起, 写作 $R \bowtie_{\text{条件}} S$

等值连接是指条件为属性 $RA=SB$

自然连接是指条件为属性 $RA=SA$ , 并且要去掉重复列, 写作 $R \bowtie S$

悬空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

外连接是指保留悬空元组的连接, 不匹配的位置填NULL, 写作 $\bowtie$

左外连接是指只保留R中悬空元组的连接, 写作 $\bowtie$

右外连接是指只保留S中悬空元组的连接, 写作 $\bowtie$

除 $\div$ : 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	$R \div S$
A B C	B C D	A
$a_1$ <u><math>b_1</math></u> $c_2$	<u><math>b_1</math></u> $c_2$ $d_1$	$a_1$
$a_2$ $b_3$ $c_1$	<u><math>b_2</math></u> $c_1$ $d_1$	
$a_3$ $b_4$ $c_6$	<u><math>b_2</math></u> $c_3$ $d_2$	
$a_1$ <u><math>b_2</math></u> $c_3$		
$a_4$ $b_6$ $c_6$		
<u><math>a_2</math> <math>b_2</math> <math>c_3</math></u>		
$a_1$ <u><math>b_2</math></u> $c_1$		

$a_2$ 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  $\pi$  \_\_\_\_ ( $\sigma$  \_\_\_\_ (表名  $\bowtie$  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  $\neg$   $\wedge$   $\vee$   $\leftrightarrow$   $\geq$   $\leq$   $>$   $<$   $=$ 。若有多个表, 表之间常用自然连接。

不等于  
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, 再对生成的子表进行除运算。

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:  $\pi_{\text{学号}, \text{课程}}(A) \div \pi_{\text{课程}}(B)$

(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。

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

## 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 $\theta$

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 $\theta$

(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 $\exists$ 和 $\forall$ 。其中 $\exists$ 的优先级大于 $\forall$

3、逻辑运算符: 包括 $\neg$ 和 $\wedge$ 和 $\vee$ 。其中 $\neg$ 的优先级大于 $\wedge$ ,  $\wedge$ 的优先级大于 $\vee$

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

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

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

$$\pi_{i_1, i_2, \dots, i_k}(R) = \{t^{(k)} \mid (\exists u) (R(u) \wedge t[1]=u[i_1] \wedge \dots \wedge t[k]=u[i_k])\}$$

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 $i_1$ ), 最后一列 $k$ 列为 $R$ 中的 $i_k$ 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 $S_{15}$ ，其中的元组 $t$ 满足条件： $t$ 属于Student表且第五列为15。

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。

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

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Student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。

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

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：

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

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

冲突可串行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锁

表示方法：上锁Xlockl ) 释放锁Unlockl )

(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S锁，但不能添加X锁

表示方法：上锁Slockl ) 释放锁Unlockl )

(3) 封锁协议

一级封锁协议：写前加写锁，事务结束释放写锁；可防止丢失修改

二级封锁协议：写前加写锁，读前加读锁，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）：写前加写锁，读前加读锁，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，都是可串行化的调度

#### (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行化的充分条件，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 will 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