
Grow

求候选码：

①只存在于左边的一定是候选码

②求闭包

左边能推出全集 ✓

不能

按个加"两边"，按个都行，再组合，但一直一定有左边！

要是有没出现在关系式中的（外部属性），一定要加上！！！

判断无关属性：

在F中，把目标式子改为，成F'

F→F'成功，则无关。

CH→D.

没H无关 求C+能推出D ✓

就分别求2个的闭包。

哪个的闭包能推出右边。

另一个无关。

哪个无关

把它去掉

求其他所有的闭包。

AD→H

A无关 (CD)+

求正则覆盖：

①化简（有无相同左部）。

②先看左边再看右。只看一边有多个的，一按一不闭。

eg. AB→C 分别判断 A.B 是否为无关属性。

③若有无关属性，则更改F'，活用F'来推

3NF

(1) 判断：

$\alpha \to \beta$ 平凡的函数依赖　　$AB \to A$，$AB \to B$

α 是 R 的一个超码。

✓ $\beta - \alpha$ 中的每个属性 A 都包含于一个候选码。

因为 α、β 无交集，$\beta - \alpha = \beta$。

（2）函数依赖集F中，所有的右部属性都在某个候选码的属性当中，显然是3NF。

(2) 3NF分解：

① 求F的正则覆盖 F_c　左右都看，别忘了！

② 求候选码　F和!!!

③ 将每个对　各成一组。

若已包含候选码 → ok

不包含 → 加上一组，为候选码。

完全包含

例：$F_c = \{A \to B, BL \to D, C \to A\}$

$R_1 = \{AB\}$　$R_2 = \{BLD\}$

$R_3 = \{AL\}$

$R_4 = \{$候选码$\}$

BCNF 判断：

$\alpha \to \beta$ 平凡依赖

α 是 R 的一个超码　就是 α 为候选码

BCNF 分解：

① 对候选码

② 右部为候选码的忽略

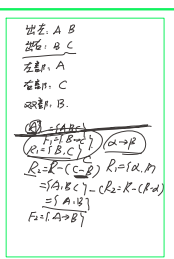
③ 看不满足②的部分

$\alpha \to \beta$

$R_1 = \{\alpha, \beta\}$

$R_2 = R - (\beta - \alpha)$

④ 看这把 R_1、R_2 是不是不是再分。



无损连接分解.

① 将题分出的2个 R_1 R_2 取 $R_1 \cap R_2$

② 看 $R_1 \cap R_2$ 与F 能不能推出 R_1或R_2

- 能：无损
- 不能：有损.

在分解时保持依赖.

①对F中每个 $\alpha \to \beta$

result = α

对于每个分解后的 R_i $t = (result \cap R_i)^+ \cap R_i$

result = result $\cup$ t.

直到 result 没变化.

$F' \cup [(F' \cap R_i)^+ \cap R_i] = F'$

直到 F' 没变化

判断可串行调度.

注意 r → w

冲突

① 按每个字母，在表格上画冲突关系

RW WW WR

② 按每个字母写顺序，画图

③ 有环 不是冲突可串行化调度

无环 满足先后顺序即可.

视图.

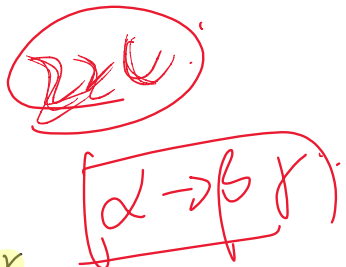
> 按每个字母分析.
> read前的 write 要顺序执行.
> 先写的先执行，后读的后执行.
> r w w (w)　　(r) r r w
> 保　　保.

只保证读的结果和最后写的结果不变.

Armstrong公理

- 自反律：$\beta \subseteq \alpha$ 则 $\alpha \to \beta$
- 增补律：$\alpha \to \beta$ 且 γ 为一属性集 $\alpha\gamma \to \beta\gamma$
- 传递律：$\alpha \to \beta$ 且 $\beta \to \gamma$ 则 $\alpha \to \gamma$.

- 合并律：$\alpha \to \beta$ $\alpha \to \gamma$ 则 $\alpha \to \beta\gamma$
- 分解律：$\alpha \to \beta\gamma$ 则 $\alpha \to \beta$ 且 $\alpha \to \gamma$
- 伪传递律：$\alpha \to \beta$ 且 $\beta\gamma \to \delta$ 则 $\alpha\gamma \to \delta$



关系代数

	（附加）	（扩展）
σ 选择	∩ 交	广义投影
Π 投影	⋈ 连接	聚集
∪ 并	÷ 除	外连接 ⟗
− 差	← 赋值	删除 $r \leftarrow r - E$
× 积（笛）		插入 $r \leftarrow r \cup E$
ρ 换名		更新

三、[18 分，每小题 3 分]

7、查询所有在 London 支行没有贷款帐户的客户编号

$\Pi_{customer\_id}(customer) - \Pi_{customer\_id}\sigma_{branch\_name="London"}(branch \bowtie loan \bowtie borrower)$

8、查询在 Brooklyn 城市的每个支行都有存款帐户的客户编号

$\Pi_{customer\_id,\ branch\_name}(account \bowtie depositor) \div \Pi_{branch\_name}\sigma_{branch\_city="Brooklyn"}(branch)$

9、查询 Brooklyn 城市各支行的存款帐户额最大值。

${}_{branch\_name}\mathcal{G}_{max(balance)}(\sigma_{branch\_city="Brooklyn"}(branch \bowtie account))$

10、将存款帐号为 A001 的所有相关存款信息从数据库中删除

$account \leftarrow account - \sigma_{account\_nnumber="A001"}(account)$

$depositor \leftarrow depositor - \sigma_{account\_nnumber="A001"}(depositor)$

11、给 Perryridge 支行所有贷款客户提供额度为 200 的存款账户。该存款账户的账号为相应贷款客户的贷款帐号。

$r_1 \leftarrow (\sigma_{branch\_name="Perryridge"}(borrower \bowtie loan))$

$account \leftarrow account \cup \Pi_{loan\_number,\ branch\_name,\ 200}(r_1)$

$depositor \leftarrow depositor \cup \Pi_{customer\_id,\ loan\_number}(r_1)$

12、删除 Perridge 支行的所有存款记录。

$account \leftarrow account - \sigma_{branch\_name="Perryridge"}(account)$

四、[24 分] 关系表结构与上题相同，试用 SQL 语句表达（其中查询语

①看最后要得到什么，就Π什么。记得在后面加上（ρ）（哪个表）。

②有"全部、所有、到们（每个）"之类的，用"÷"

÷前面的Π中："全部"之后的词 与 最后要求的词。

要写全，用到了的都要写。

SQL

Π ~~~ σ ~~~ (~~~~~)

select　　where　　from.

（3）使用关系代数和SQL语句找出账户平均余额小于5000元的支行，显示支行名称及账户平均余额

$$\sigma_{ab<5000}(\rho_{branch-balance(branch-name,ab)}({}_{branch-name}\mathcal{G}_{avg(balance)}(account)))$$

```
select branch-name, avg(balance)
from account
group by branch-name
having avg(balance)<5000
```

（4）使用关系代数和SQL语句找出所有在银行中有贷款但无账户的客户。

$$\prod_{customer-name}(borrower)-\prod_{customer-name}(depositor)$$

```
select distinct customer-name
from borrower
where customer-name not in
      (select customer-name from depositor)
```

（5）使用关系代数和SQL语句对所有存款余额大于平均存款额的账户增加3%的利息。

$$account \leftarrow \prod_{account-number,branch-name,balance*1.03}(\sigma_{balance>avg(balance)}(account))$$
$$\cup \prod_{account-number,branch-name,balance}(\sigma_{balance\leq avg(balance)}(account))$$

```
update account
set balance=balance*1.03
where balance>(select avg(balance) from account)
```

（3）使用关系代数和SQL语句找出账户平均余额小于5000元的支行，显示支行名称及账户平均余额

$$\sigma_{ab<5000}(\rho_{branch-balance(branch-name,ab)}({}_{branch-name}g_{avg(balance)}(account)))$$

```
select branch-name, avg(balance)
from account
group by branch-name
having avg(balance)<5000
```

SQL

(1) select
from
where

(2) update 哪个
set 把什么改成什么
where (让将变了，此表的第一个属性).

(3)
select A, avg(B) 取平均
from 从哪
Group by A 为A新建表，便于排序
Order by A desc A降序.

(4) update
set A = A* ~

聚集函数 分组聚集.

(5) create table 名(
ID char(1) primary key,
primary key(—, —).
); 记得标主键.

(6) 更新表结构.
alter table — ADD —
drop —
drop table —

(7) 更新内容.
insert into — values (1,2,3…)
Delete from 删元组
update 谁 set 变什么 修改元组.

(8)

```
select name, course_id
from instructor , teaches
where instructor.ID = teaches.ID
and   instructor. dept_name =
'Art'
```

为值等于.

(9) 字符串匹配.

Select
from
where — like

'%dar%' → ##darAA
'dar\' → darA
'dar%' → darAA.

(10) 集合运算.

union ∪
intersect ∩
except −

(Select — from — where)
union
(Select — from — where)

相同或相容的活动.

(11) count(*) 计数

```
select count (*)
from course;
```

(12)

avg: average value
min: minimum value
max: maximum value
sum: sum of values
count: number of values

(13)

(1) 单一关系上的约束
- not null 非空
- primary key 主键
- unique 唯一约束
- check (P), where P is a predicate

```
create table section (
    course_id varchar (8),
    sec_id varchar (8),
    semester varchar (6),
    year numeric (4,0),
    building varchar (15),
    room_number varchar (7),
    time_slot_id varchar (4),
    primary key (course_id, sec_id, semester, year),
    check (semester in ('Fall', 'Winter', 'Spring', 'Summer'))
);
```

在关系模式下面写.

(2) 参照完整性约束

```
create table course (
    course_id    char(5) primary key
    title        varchar(20),
    dept_name varchar(20) references department
)
```

(14) 视图.

可以是组 select from 语句.

create view — as < >

视图名 查的东西

(15) 关系测试.

exists
not exists

判断查询的返回值是否为空.

(16)

- in 交集

```
select distinct course_id
from section
where semester = 'Fall' and year= 2009 and
      course_id in (select course_id
                    from section
                    where semester = 'Spring' and year= 2010);
```

- not in 差集

```
select distinct course_id
from section
where semester = 'Fall' and year= 2009 and
      course_id not in (select course_id
                        from section
                        where semester = 'Spring' and year= 2010);
```

select: allows read access to relation,or the ability to query using the view

- Example: grant users U_1, U_2, and U_3 **select** authorization on the *instructor* relation:

  grant select on *instructor* **to** U_1, U_2, U_3

insert: the ability to insert tuples

update: the ability to update using the SQL update statement

delete: the ability to delete tuples.

all privileges: used as a short form for all the allowable privileges

The **revoke** statement is used to revoke authorization.

revoke <privilege list>

on <relation name or view name> **from** <user list>

Example:

revoke select on *branch* **from** U_1, U_2, U_3

where — Is null

0.

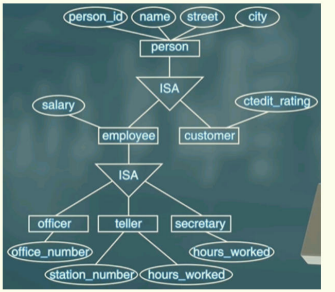
```sql
CREATE VIEW V AS
SELECT Mid,MN

FROM M

WHERE q_Time-今日日期=3

```

5.用SQL语句，修改物料名为"海绵"的库存，增加100件

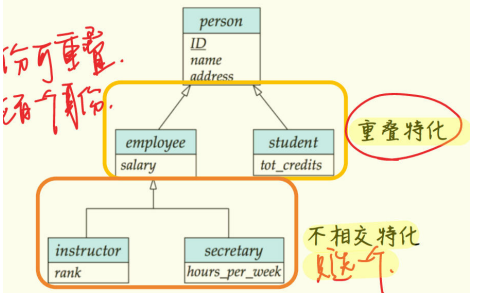
```sql
UPDATE st
SET SNum=CASE
    WHEN st.Mid=(SELECT Mid FROM m WHERE MN='海绵') THEN
        SNum+100
    ELSE
        SNum
END ;
```

ER

- 矩形代表实体集
- 菱形代表联系集
- 双菱形代表连接到弱实体集的标志性联系集
- 线段将实体集连接到联系集
- 虚线将联系集属性连接到联系集
- 双线显示实体在联系集中的参与度
- 椭圆代表属性(或者是矩形里面的项代表属性)
  - 双线椭圆代表多值属性
  - 用虚线画的椭圆代表派生属性
- 下划线代表主键



全部参与.



实验课程（课程id，上课名，上课时间）

选择（课程id，学生id）

学生（学生id，学生名，学生电话，教师id）

指导教师（教师id，教师名，教师电话）

参与（课程id，教师id）

为了保证全参与，可以使用断言实现，因为无法在设计中表示，选择和学生之间的全参与可以不写。

理财项目(项目id，项目名，项目计划) 主键项目id

买卖信息(买卖id，买时间，卖时间，买卖金额)主键买卖id

客户(客户id，经理id，客户名，客户电话)主键客户id和经理id，外键经理id参考理财经理

理财经理(经理id，经理名，经理电话)主键经理id

购买(项目id，客户id，买卖id)主键项目id客户id买卖id外键也是它们，参考理财项目、买卖信息和客户

发展客户(客户id，下线id)对下线id有unique约束，主键是客户id下线id