

数据库复习

- create
 - 建表
- drop
 - 删表
 - 删视图
 - 删触发器
- select
 - group by
 - order by(asc&desc&top)
 - exists
 - any&all
- 集合操作
 - Union
- 数据更新
 - 插入数据
 - 删除数据
 - 修改数据
- 视图
 - 新建视图
- 授权
- 回收
- 数据库完整性
 - 实体完整性
 - 参照完整性
 - 违约删除
 - 违约插入
 - 用户定义的完整性
- 触发器
 - 建立触发器
- 规范化
 - 1NF
 - 2NF
 - 3NF
 - BCNF
 - 4NF
- 数据依赖的公理系统

- Armstrong公理系统
 - 自反律
 - 增广律
 - 传递律
 - 合并规则
 - 伪传递规则
 - 分解规则
 - 引理6.1
- 计算 $X_{\{F\}}^+$ 闭包
 - 引理6.3
- T-SQL
 - 游标
 - 存储过程
 - 不带参数
 - 带参数

create

建表

- 将A作为主键

SQL

```
CREATE TABLE Table1
(
  A INT PRIMARY KEY,
  B CHAR(10),
)
```

- 将(A,B)作为主键

SQL

```
CREATE TABLE Table1
(
  A INT,
  B CHAR(10),
  PRIMARY KEY (A, B)
)
```

- 设置外键约束

SQL

```
CREATE TABLE Table2
(
  A INT,
  B CHAR(10),
  FOREIGN KEY(A) REFERENCES Table1(A)
)
```

drop

删表

SQL

```
DROP TABLE Table1
```

删视图

SQL

```
DROP VIEW IS_S1
```

删触发器

SQL

```
DROP TRIGGER T1
```

select

group by

如果在返回集字段中，这些字段要么就要包含在Group By语句的后面，作为分组的依据；要么就要被包含在聚合函数中

--错误写法

```
SELECT sno,
       cno
FROM   student
GROUP BY sno
```

--正确写法

```
SELECT sno,
       Count(cno)
FROM   student
GROUP BY sno
```

- 在SC表中查询选了课程号为'1','2','3'且平均成绩在90分以上的学生学号和这几门课的平均成绩

```
SELECT sno,
       Avg(grade)
FROM   SC
WHERE  cno IN ( '1', '2', '3' )
GROUP BY sno
HAVING Avg(grade) >= 90
```

order by(asc&desc&top)

- 查询其他系中比信息系所有学生年龄都小的学生里年龄最小的两个学生姓名及年龄。

```
SELECT TOP 2 sname,
            sage
FROM   student
WHERE  sage < ALL (SELECT sage
                  FROM   student
                  WHERE  sdept = 'IS')
      AND sdept <> 'IS'
ORDER BY sage asc
```

- 查询全体学生情况，查询结果按所在系的系号升序排列，同一系中的学生按年龄降序排列。

```
SELECT *
FROM student
ORDER BY sdept ASC,
        sage DESC
```

- 与group by结合

查询从2018年1月1日到2018年5月31日期间该网店已经销售且销量最低的食品的销售记录，包括食品编号、食品名称、销售总数量

食品表F(Fno,Fname,Fprice,Fdate,Fquality)
 顾客表c(Cno,Cname,Csex,Cage)
 销售表s(Sno,Fno,Cno,Scount,Ssum,Sdate)

```
SELECT TOP 1 Fno,
              Fname,
              Sum(Ssum)
FROM s,
     F
WHERE s.Fno = F.Fno
      AND Sdate BETWEEN '2018年1月1日' AND '2018年5月31日'
GROUP BY Fno,
          Fname
ORDER BY Sum(Ssum) ASC
```

exists

- 查询所有选修了2号课程的学生姓名。

```
select Sname
from Student
where exists(
    select *
    from SC
    where SC.Sno=Student.Sno and SC.Cno='2'
)
```

- 查询没有选修1号课程的学生姓名。

SQL

```
select Sname
from Student
where not exists(
    select *
    from SC
    where SC.Sno=Student.Sno and SC.Cno='1'
)
```

- 查询与"刘晨"在同一个系学习的学生信息(学生号、学生姓名、系别)。可以用带EXISTS谓词的子查询替换:

SQL

```
select Sno,Sname,Sdept
from Student S1
where exists(
    select *
    from Student S2
    where S1.Sdept=S2.Sdept and S2.Sname='刘晨'
)
```

- 查询选修了全部课程的学生姓名。
查询这样的学生，没有一门课程是他不选修的

SQL

```
select Sname
from Student
where not exists(
    select *
    from Course
    where not exists(
        select *
        from SC
        where Student.Sno=SC.Sno and Course.Cno=SC.Cno
    )
)
```

- 查询至少选修了学生95002选修的全部课程的学生号码。
查询这样的学生，不存在这样的课程y，95002选了但是该学生没有选

SQL

```
select Sno
from Student
where not exists(
    select *
    from SC SC1
    where SC.Sno='95002' and not exists(
        select *
        from SC SC2
        where SC2.Sno=Student.Sno and SC2.Cno=SC1.Cno
    )
)
```

- eg1.
查找销售日期在2018年1月1日至2018年3月1日期间，购买了所有单价大于50食品男性顾客的顾客名称、顾客性别和顾客年龄

食品表F(Fno,Fname,Fprice,Fdate,Fquality)
顾客表C(Cno,Cname,Csex,Cage)
销售表s(Sno,Fno,Cno,Scount,Ssum,Sdate)

SQL

```
SELECT Cname,
       Csex,
       Cage
FROM   C
WHERE  Csex = '男'
      AND NOT EXISTS(
        SELECT *
        FROM   F
        WHERE  Fprice > 50
              AND NOT EXISTS(
                SELECT *
                FROM   S
                WHERE  Sdate BETWEEN '2018年1月1日' AND '2018年3月1日'
                      AND S.Fno = F.Fno
                      AND S.Cno = C.Cno))
```

- eg2.

查找下单日期在2019年4月1日（含）至2019年5月31日（含）期间，购买了所有服装材质为“纯棉”的女性顾客的顾客信息，包括顾客姓名、顾客性别、顾客年龄、顾客电话、顾客地址。

服装表G(Gno,Gname,Gsize,Gcolor,Gprice,Gmaterial)
 顾客表C(Cno,Cname,Csex,Cage,Cphone,Caddress)
 订单表O(Ono,Gno,Cno,Ocount,Osum,Odate)
 退货表B(Bno,Ono,Gno,Cno,Bcount,Bsum,Bdate)
 库存表GW(Gno,Wno,num,Ttime)

SQL

```
SELECT Cname,Csex,Cage,Cphone,Caddress
FROM C
WHERE Csex = '女'
      AND NOT EXISTS(
        SELECT *
        FROM G
        WHERE Gmaterial = '纯棉'
              AND NOT EXISTS (
                SELECT *
                FROM O
                WHERE Odate BETWEEN '2019年4月1日' AND '2019年5月
31日'
                      AND O.Cno = C.Cno
                      AND O.Gno = G.Gno))
```

any&all

- any

查询其他系中比信息系所有学生年龄都小的学生姓名及年龄

SQL

```
SELECT sname,
       sage
FROM student
WHERE sage < ALL (SELECT sage
                  FROM student
                  WHERE sdept = 'IS')
      AND sdept <> 'IS'
```


- all

查询其他系中比信息系某一学生年龄小的学生姓名及年龄。

SQL

```
SELECT sname,
       sage
FROM   student
WHERE  sage < ANY (SELECT sage
                   FROM   student
                   WHERE  sdept = 'IS')
AND    sdept <> 'IS'
```

集合操作

Union

- 查询学校中所有师生的姓名。

SQL

```
SELECT SnameFROM Student
UNION
SELECT TnameFROM Teacher;
```

数据更新

插入数据

- 插入单个元组

将一个新学生记录（学号：95020；姓名：陈冬；性别：男；所在系：IS；年龄：18岁）插入到Student表中。

SQL

```
insert into Student values()
```

插入一条选课记录('95020', '1 ')。

SQL

```
INSERT INTO SC(Sno, Cno)VALUES (' 95020 ', ' 1 ');
```

- 插入子查询结果

```
INSERT INTO Deptage(Sdept, Avgage)
SELECT Sdept, AVG(Sage)
FROM Student
GROUP BY Sdept;
```

删除数据

- 删除Student表中信息系的学生信息

```
delete from Student
where Sdept='IS'
```

- 删除信息系所有学生的选课记录

```
DELETE FROM SC
WHERE SC.Sno IN (SELECT Student.Sno
                  FROM Student
                  WHERE Sdept = 'IS')
```

修改数据

- 将学生95001的年龄改为22岁

```
UPDATE Student SET
set Sage=22
WHERE Sno=' 95001 ';
```

- 对食品表F中的食品单价进行调整，将被所有顾客购买、食品单价大于50且总销售金额最高的前3件食品分别降价5元。

食品表F(Fno,Fname,Fprice,Fdate,Fquality)
 顾客表c(Cno,Cname,Csex,Cage)
 销售表s(Sno,Fno,Cno,Scount,Ssum,Sdate)

```

UPDATE F
SET    Fprice = Fprice - 5
WHERE  Fno IN(
SELECT TOP 3 Fno
FROM    S
WHERE   s.fno IN(SELECT fno
                  FROM    F f1
                  WHERE   Fprice > 50
                  AND NOT EXISTS(
                    SELECT *
                    FROM    C
                    WHERE   NOT EXISTS(
                      SELECT *
                      FROM    S s1
                      WHERE   f1.Fno = s1.Fno
                              AND C.Cno = s1.Cno)))
GROUP  BY Fno
ORDER  BY Sum(Ssum) DESC)

```

视图

新建视图

- 建立信息系学生的视图

```

create view V1
as
select *
from Student
where Sdept='IS'

```

- 建立选修了1号课程的学生视图(包括学号、姓名、成绩)

```
create view v2(Sno,Sname,grade)
as
select Sno,Sname,grade
from SC,Student
where SC.Sno=Student.Sno and SC.Cno='1'
```

- 定义一个反映学生出生年份的视图（包括学号、姓名、出生年份）。
设置一些派生属性列, 也称为虚拟列--Sbirth带表达式的视图必须明确定义组成视图的各个属性列名

```
CREATE VIEW V3(Sno,Sname,Sbirth)
as
select Sno,Sname,2024-Sage
from Student
```

查询视图、删除视图与普通表相同, 不再多说

对视图的更新都会改为对基本表的更新

授权

- 把查询Student表权限授给用户U1

```
grant select
on student
to U1
```

- 把对Student表的全部权限授予用户U2和U3

```
grant all privileges
on Student
to U2,U3
```

- 把对表SC的查询权限授予所有用户

SQL

```
grant select
on SC
to public
```

- 把查询Student表和修改学生姓名的权限授给用户U4

SQL

```
grant update(Sname), select
on Student
to U4
```

- 把对表SC的INSERT权限授予U5用户，并允许他再将此权限授予其他用户

SQL

```
grant insert
on SC
to U5
with grant option
```

回收

- 把用户U4修改学生学号的权限收回

SQL

```
revoke update(Sno)
on Student
from U4
```

- 把用户U5对SC表的INSERT权限收回

SQL

```
revoke insert
on SC
from U5 cascade--级联收回
```

若U5有将此权限授予给U3

若没有增加cascade关键字，会报错

增加cascade级联关键字后，会一并将U3 U5的此权限收回

数据库完整性

实体完整性

实体完整性规则：若属性A是基本关系R的主属性，则属性A不能取空值

参照完整性

若表A中的外码A1是参照表B中的主码B1，则A1要么取

- 空值
- 等于B中某个元组的主码值

违约删除

出现违约操作的情形：

删除被参照关系的某个元组（**student**）

而参照关系有若干元组(**SC**)的外码值与被删除的被参照关系的主码值相同

可有三种策略

- 受限删除（NO ACTION）
当参照关系中没有任何元组的外码值与要删除的被参照关系的元组的主码值相对应时，系统才执行删除操作，否则拒绝此删除操作
- 级联删除（CASCADE）
将参照关系中外码值与被参照关系中要删除元组主码值相对应的元组一起删除

SQL

```
CREATE TABLE SC(  
  Sno CHAR(5)foreign key references Student(Sno)  
  ON DELETE CASCADE,  
  Cno CHAR(1) ,  
  Grade INT  
);
```

- 置空值删除（NULLIFIES）
删除被参照关系的元组，并将参照关系中与被参照关系中被删除元组主码值相等的外码值置为空值

违约插入

出现违约操作的情形

- 需要在参照关系中插入元组，而被参照关系不存在相应的元组

可有两种策略

- 受限插入
仅当被参照关系中存在相应的元组，其主码值与参照关系插入元组的外码值相同时，系统才执行插入操作，否则拒绝此操作
- 递归插入
首先向被参照关系中插入相应的元组，其主码值等于参照关系插入元组的外码值，然后向参照关系插入元组。

用户定义的完整性

约束命名

Constraint <约束名>

[Primary key... | Foreign Key... | Check ...]

- NOT NULL
- UNIQUE
- CHECK

触发器

通过触发器来定义复杂的完整性规则

建立触发器

SQL

```
create trigger T1
on Table1
as
```

- 学生成绩不低于60分,低于60分自动赋为60分

```
create trigger T1
on SC
for insert
as
update SC
set grade=60
where exists(
    select *
    from inserted
    where inserted.grade<60 and SC.Sno=inserted.Sno and
SC.Cno=inserted.Cno
)
```

- 计算机 (sdept为'CS') 学生成绩不低于60分,低于60分自动赋为60分

```
create trigger T2
on SC
for insert
as
update SC
set grade=60
where exists(
    select *
    from inserted,Student
    where inserted.Sno=Student.Sno and sdept='SC'and
inserted.grade<60 and SC.Sno=inserted.Sno and SC.Cno=inserted.Cno
)
```

- 数学是必修课, 所有学生都必须选, 学生退学则将成绩置0
当往student表中插入学生, 就立即向SC表插入选课记录


```
create trigger T1
on Student
for insert
as
insert into SC(Sno,Cno)
select Sno,'2'
from inserted
```

当从student表中删除学生，连同将sc表的成绩置0

```
create trigger T1
on Student
for delete
as
update SC
set grade=0
where exists(
    select *
    from deleted
    where deleted.Sno=SC.Sno
)
```

- 创建触发器，当更新student表中的学号时，也同时更新sc表中的学号

```
create trigger T1
on Student
for update
as
update SC
set SC.sno=inserted.sno
from SC,inserted,deleted
where SC.Sno=deleted.Sno
```

- 创建一个触发器，当删除student表中的某条记录时，也同时删除sc表中的记录

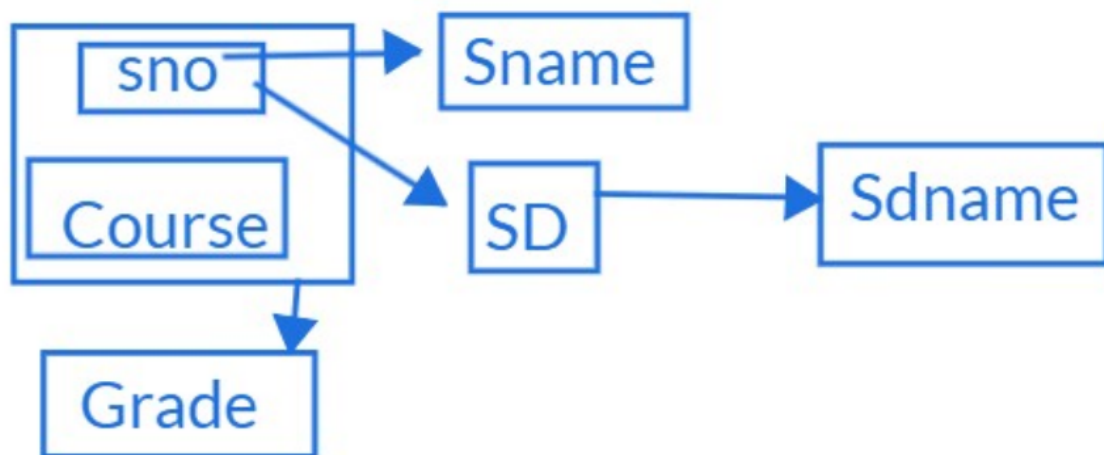
```
create trigger T1
on Student
for delete
as
delete from SC,deleted
where SC.Sno=deleted.Sno
```

规范化

$$1NF \supset 2NF \supset 3NF \supset BCNF \supset 4NF \supset 5NF$$

1NF

已知学生关系模式S(Sno,Sname,SD,Sdname,Course,Grade)其中, Sno是学号, Sname是姓名, SD是系名, Sdname是系主任名, Course是课程, Grade是成绩



- 基本函数依赖:
 - $Sno \rightarrow Sname$
 - $Sno \rightarrow SD$
 - $SD \rightarrow Sdname$
 - $(Sno, Course) \rightarrow Grade$
- 主码: (Sno,Course)

2NF

定义: 若关系模式 $R \in 1NF$, 并且每一个非主属性都完全函数依赖于 R 的码, 则 $R \in 2NF$ 。

1NF:

$$R(U) \in 1NF$$

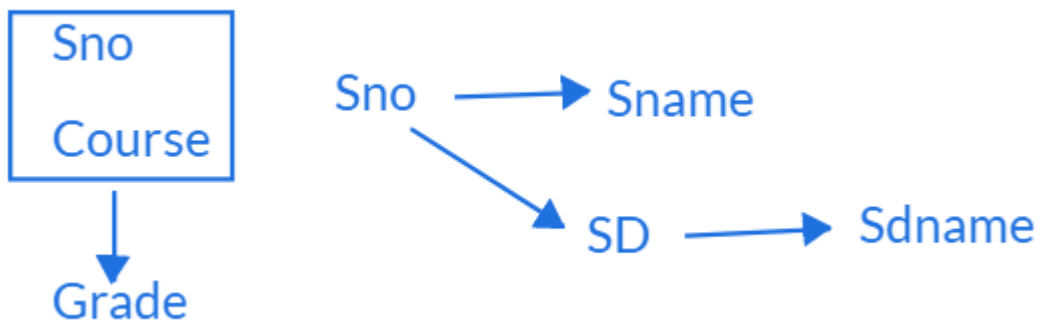
$$R(U) \notin 2NF$$

- 基本函数依赖:
 - $Sno \rightarrow Sname$
 - $Sno \rightarrow SD$
 - $SD \rightarrow Sdname$
 - $(Sno, Course) \rightarrow Grade$
- 主码: (Sno, Course)
可以看出其中Sname、SD是非主属性部分函数依赖于R的码。
将其拆分。

2NF:

$$SC(Sno, Course) \in 2NF$$

$$SSDS(Sno, SD, Sdname, Sname) \in 2NF$$



3NF

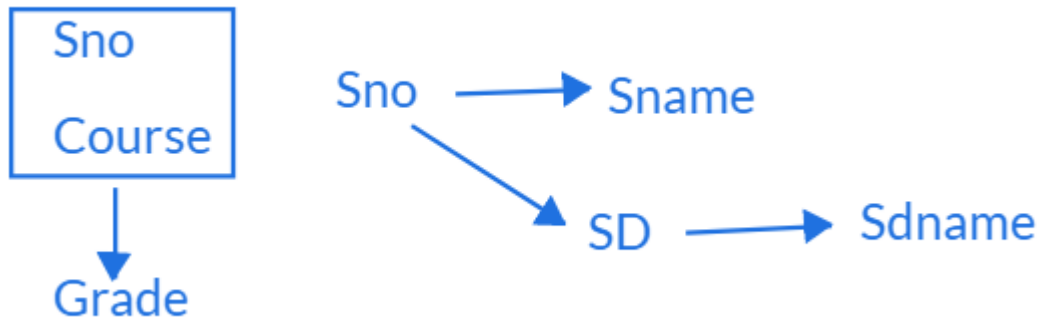
定义6.8 关系模式 $R\langle U, F \rangle$ 中若不存在这样的码 X 、属性组 Y 及非主属性 Z ($Z \not\subseteq Y$), 使得 $X \rightarrow Y$, $Y \not\rightarrow X$, $Y \rightarrow Z$, 成立, 则称 $R\langle U, F \rangle \in 3NF$ 。

即不存在非主属性对码的传递函数依赖。

2NF:

$$SC(Sno, Course) \in 2NF$$

$$SSDS(Sno, SD, Sdname, Sname) \in 2NF$$



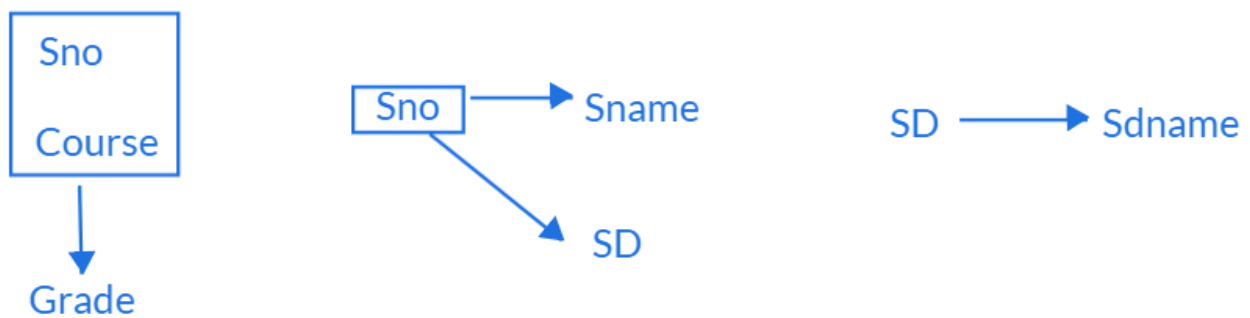
其中Sno为码。存在 $Sno \rightarrow SD$, $SD \rightarrow Sdname$, 且 $SD \nrightarrow Sno$ and $Z \not\subseteq SD$ 。因此不为 3NF

3NF:

$$SC(Sno, Course) \in 3NF$$

$$SDS(Sno, SD, Sname) \in 3NF$$

$$SS(Sno, Sdname) \in 3NF$$



不存在非主属性对码的传递函数依赖。

BCNF

定义6.9 设关系模式 $R \langle U, F \rangle \in 1NF$, 如果对于 R 的每个函数依赖 $X \rightarrow Y$, 若 Y 不属于 X , 则 X 必含有码, 那么 $R \in BCNF$ 。

没有任何属性完全函数依赖于非码的任何一组属性

- 证明题: 若关系模式 $R \in BCNF$, 则 $R \in 2NF$ 。

反证法:

若 $R \notin 2NF$, 即存在非主属性 Y , 主码 X , 有 $X \xrightarrow{p} Y$

即存在码X的子集 $X^t \xrightarrow{F} Y$ ，又由于 X^t 不含码，与BCNF的定义相悖。
因此可证若关系模式 $R \in \text{BCNF}$ ，则 $R \in 2\text{NF}$

4NF

定义6.10 关系模式 $R \langle U, F \rangle \in 1\text{NF}$ ，如果对于R的每个非平凡多值依赖 $X \twoheadrightarrow Y$ ($Y \not\subseteq X$)，X都含有候选码，则 $R \in 4\text{NF}$ 。

- 没有对平凡多值依赖有任何要求。
- 对非平凡：要么没有多值依赖，要么要求X含有主码

• 关系模式规范化的基本步骤

消除决定属性集非码的非平凡函数依赖	1NF
	↓ 消除非主属性对码的部分函数依赖
	2NF
	↓ 消除非主属性对码的传递函数依赖
	3NF
	↓ 消除主属性对码的部分和传递函数依赖
	BCNF
	↓ 消除非平凡且非函数依赖的多值依赖
	4NF

数据依赖的公理系统

Armstrong公理系统

关系模式 $R \langle U, F \rangle$ 来说有以下的推理规则：

自反律

若 $Y \subseteq X \subseteq U$ ，则 $X \rightarrow Y$ 为F所蕴含。

证明：

对于 $R \langle U, F \rangle$ 中任一关系r的任意两个元组t,s

若 $t[X] = s[X]$ ，由于 $Y \subseteq X$ ，有 $t[Y] = s[Y]$

根据函数依赖的定义可证 $X \rightarrow Y$

增广律

若 $X \rightarrow Y$ 为 F 所蕴含, 且 $Z \subseteq U$, 则 $XZ \rightarrow YZ$ 为 F 所蕴含

证明:

对于 $R \langle U, F \rangle$ 中任一关系 r 的任意两个元组 t, s

若 $t[XZ] = s[XZ]$, 则有 $t[X] = s[X]$ 、 $t[Z] = s[Z]$

又由于 $X \rightarrow Y$, 已知 $t[X] = s[X]$, 则有 $t[Y] = s[Y]$

因此 $t[YZ] = s[YZ]$

根据函数依赖的定义可证 $XZ \rightarrow YZ$

传递律

若 $X \rightarrow Y$ 及 $Y \rightarrow Z$ 为 F 所蕴含, 则 $X \rightarrow Z$ 为 F 所蕴含

证明:

对于 $R \langle U, F \rangle$ 中任一关系 r 的任意两个元组 t, s

由于 $X \rightarrow Y$, 若 $t[X] = s[X]$, 有 $t[Y] = s[Y]$

由于 $Y \rightarrow Z$, 已知 $t[Y] = s[Y]$, 有 $t[Z] = s[Z]$

根据函数依赖的定义可证 $X \rightarrow Z$

合并规则

由于 $X \rightarrow Y$, $X \rightarrow Z$, 有 $X \rightarrow YZ$

证明:

对于 $R \langle U, F \rangle$ 中任一关系 r 的任意两个元组 t, s

由于 $X \rightarrow Y$, 若 $t[X] = s[X]$, 有 $t[Y] = s[Y]$

由于 $X \rightarrow Z$, 已知 $t[X] = s[X]$, 有 $t[Z] = s[Z]$

得证 $X \rightarrow YZ$

伪传递规则

由于 $X \rightarrow Y$, $WY \rightarrow Z$, 有 $WX \rightarrow Z$

证明:

对于 $R \langle U, F \rangle$ 中任一关系 r 的任意两个元组 t, s

由于 $X \rightarrow Y$, 若 $t[X] = s[X]$, 有 $t[Y] = s[Y]$

由于 $WY \rightarrow Z$, 已知 $t[Y] = s[Y]$, 若 $t[W] = s[W]$, 有 $t[Z] = s[Z]$

得证: 若 $t[X] = s[X]$, $t[W] = s[W]$, 有 $t[Z] = s[Z]$

即 $XW \rightarrow Z$

分解规则

由于 $X \rightarrow Y$, 及 $Z \subseteq Y$, 有 $X \rightarrow Z$

证明:

由于 $X \rightarrow Y$, 若 $t[X] = s[X]$, 有 $t[Y] = s[Y]$

又由于 $Z \subseteq Y$, 有 $t[Z] = s[Z]$

因此得证 $X \rightarrow Z$

引理6.1

$X \rightarrow A_1 A_2 \dots A_k$ 成立的充要条件是 $X \rightarrow A_i$ 成立

计算 X_F^+ 闭包

例2.

例: 设有关系模式 $R(U, F)$, 其中

$U = \{A, B, C, D, E, I\}$

$F = \{A \rightarrow D, AB \rightarrow E, BI \rightarrow E, CD \rightarrow I, E \rightarrow C\}$

请计算 $(AE)_F^+$

- $X^{(0)} = AE$
- 求 $X^{(1)}$
找到 $A \rightarrow D, E \rightarrow C$
 $X^{(1)} = X^{(0)} \cup CD = ACDE$
由于 $X^{(1)} \neq X^{(0)}$ 算法继续
- 求 $X^{(2)}$
找到 $A \rightarrow D, E \rightarrow C, CD \rightarrow I$
 $X^{(2)} = X^{(1)} \cup CDI = ACDEI$
由于 $X^{(1)} \neq X^{(0)}$ 算法继续
- $X^{(3)} = X^{(2)}$, 算法停止。 $(AE)_F^+ = ACDEI$

引理6.3

引理 6.3 $F^+ = G^+$ 的充分必要条件是 $F \subseteq G^+$ ，和 $G \subseteq F^+$

证明：

必然性显然，证明充分性。

即已知 $F \subseteq G^+$ 、 $G \subseteq F^+$ ；证 $F^+ = G^+$ ；即证 $F^+ \subseteq G^+$ 、 $F^+ \supseteq G^+$

当 $F \subseteq G^+$ 时，有 $X_F^+ \subseteq X_{G^+}^+$

根据子集的定义可知，任意 $X \rightarrow Y \in F^+$ ，有 $Y \subseteq X_{F^+} \subseteq X_{G^+}^+$

即 $X \rightarrow Y \in (G^+)^+ = G^+$ ，根据子集的定义，有 $F^+ \subseteq G^+$

令一边同理可证

T-SQL

游标



课堂作业

- 鞋子表S(Sno, Sname, Sprice, Sbrand, Scolor)分别为：鞋子编号，鞋子名称，鞋子单价，鞋子品牌，鞋子颜色；
- 顾客表C(Cno, Cname, Csex, Cage, Cphone)分别为：顾客编号，顾客名称，顾客性别，顾客年龄，顾客电话；
- 订单表O(Ono, Sno, Cno, Ocount, Osum, Odate)分别为：订单号，鞋子编号，顾客编号，销售数量，销售金额（备注：不是单价），订单日期；
- 退货表R(Rno, Ono, Sno, Cno, Rcount, Rsum, Rdate)分别为：退货单号，订单号，鞋子编号，顾客编号，退货数量，退货金额，退货时间；
- 库存表I(Sno, Inum, Idate)分别为：鞋子编号，库存量，清点时间；
- 进货表P(Pno, Sno, Pnum, Pprice, Pdate)分别为：进货编号，鞋子编号，进货数量，进货单价，进货时间；
- 其中，表S由Sno唯一标识，表C由Cno唯一标识，表O由Ono唯一标识，表R由Rno唯一标识，表I由Sno唯一标识，表P由Pno唯一标识。



课堂作业

- 题目1: 网店要做一个618促销活动, 需要输出订单日期在2020-04-15 (含) 到2020-06-10 (含), 所有年龄在20到30岁的顾客的鞋子购买信息, 输出格式为“顾客编号, 顾客名称, 购买鞋子明细 (鞋子名称1|鞋子名称2|...)”, 多个鞋子名称之间以“|” 隔开 (相同的鞋子名称只能出现一次)
- 鞋子表S(Sno, Sname, Sprice, Sbrand, Scolor)
- 顾客表C(Cno, Cname, Csex, Cage, Cphone)
- 订单表O(Ono, Sno, Cno, Ocount, Osum, Odate)
- 退货表R(Rno, Ono, Sno, Cno, Rcount, Rsum, Rdate)
- 库存表I(Sno, Inum, Idate)
- 进货表P(Pno, Sno, Pnum, Pprice, Pdate)

```

declare @Cname char(10);
declare @Cno char(10);
declare @Sname char(10);
declare @output char(max);
declare cur1 cursor for
Select C.cno
from C, O
where C.cno = O.cno and Odate between '2020-04-15' and '2020-06-
10' and Cage between 20 and 30
begin
    open Cur1
    fetch next from cur1 into @cno;
    while(@@fetch_status = 0)
    begin
        declare cur2 cursor for
        select Cno, Cname, Sname from S,O,C
        where O.sno = S.sno and O.Cno = C.cno and O.cno = @cno
        begin
            open cur2
            fetch next from cur2 into @Cnon, @Cname, @Sname
            set @output = @Cno + @Cname+ '('+ '购买鞋子明细' +
@Sname

            fetch next from cur2 into @Cno, @Cname, @Sname
            while (@@fetch_status = 0)
            begin
                set @output = @output + '|' + @Sname
                fetch next from cur2 into @Cno, @Cname, @Sname
            end
            set @output = @output + ')'
            print @output
            close cur2
            deallocate cur2
        end
        fetch next from cur1 into @cno;
    end
    close cur1
    deallocate cur1
end

```

![[Pasted image 20240617004622.png]]

![[Pasted image 20240617004636.png]]

```
-- SQL
```

```
declare @Cno varchar(20);--顾客编号
declare @Cname varchar(20);--顾客名
declare @Dname varchar(20);--唱片名
declare @Isum int;--单笔消费
declare @Total int;--总消费
declare @output varchar(max);
declare cur1 cursor
for
select Cno
from C,I
where C.Cno=I.Cno and C.sex='男' and I.Idate between '2024-01-01'
and '2024-03-31'and Cage between 20 and 30
begin
    open cur1;
    fetch next from cur1 into @Cno;
    while(@@fetch_status=0)
    begin
        declare cur2 cursor
        for
        select distinct I.Cno,C.Cname,D.Dname,I.Isum
        from I,C,D
        where I.Cno=C.Cno and I.Dno=D.Dno and I.Cno=@Cno
        begin
            open cur2;
            fetch next from cur2 into @Cno,@Cname,@Dname,@Isum;
            set @Total=0;
            set @Total=Total+@sum;
            set @output=@Cno+', '+@Cname+', ('+@Dname
            fetch next from cur2 into @Cno,@Cname,@Dname,@Isum;
            while(@@fetch_status=0)
            begin
                set @output=@output+'|'+@Dname;
                set @Total=@Total+@Isum;
                fetch next from cur2 into @Cno,@Cname,@Dname,@Isum;
            end
            set @output=@output+'),'+str(@Total);
            print @output;
            close cur2;
            deallocate cur2;
        end
        fetch next from cur1 into @Cno;
    end
end
```

```

    close cur1;
    deallocate cur1;
end

```

存储过程

不带参数

SQL

```

create proc proc_get_student
as
select * from student;

```

带参数

1. 创建一个带参数的存储过程。

SQL

```

create proc proc_find_stu(
@startSno char(7),
@endSno char(7)
)
as
select * FROM Student WHERE Sno BETWEEN @startSno AND @endSno;

```

2. 创建一个带参数通配符的存储过程。

SQL

```

CREATE PROC proc_findStudentByName(
@namevarchar(20) = '%文%',
@nextName varchar(20) = '%'
AS
SELECT * FROM Student WHERE Sname LIKE @name AND Sname LIKE
@nextName;



```

3. 创建一个带输出参数存储过程。

```

CREATE PROC proc_getStudentRecord(
  @Sno CHAR(7), --默认输入参数
  @Sname CHAR(10) out, --输出参数
  @Sage INT output--输入输出参数
)AS
SELECT @Sname = Sname, @Sage = Sage from Student WHERE Sno= @Sno;

```

课堂作业

- 鞋子表S(Sno, Sname, Sprice, Sbrand, Scolor)分别为：鞋子编号，鞋子名称，鞋子单价，鞋子品牌，鞋子颜色；
- 顾客表C(Cno, Cname, Csex, Cage, Cphone)分别为：顾客编号，顾客名称，顾客性别，顾客年龄，顾客电话；
- 订单表O(Ono, Sno, Cno, Ocount, Osum, Odate)分别为：订单号，鞋子编号，顾客编号，销售数量，销售金额（备注：不是单价），订单日期；
- 退货表R(Rno, Ono, Sno, Cno, Rcount, Rsum, Rdate)分别为：退货单号，订单号，鞋子编号，顾客编号，退货数量，退货金额，退货时间；
- 库存表I(Sno, Inum, Idate)分别为：鞋子编号，库存量，清点时间；
- 进货表P(Pno, Sno, Pnum, Pprice, Pdate)分别为：进货编号，鞋子编号，进货数量，进货单价，进货时间；
- 其中，表S由Sno唯一标识，表C由Cno唯一标识，表O由Ono唯一标识，表R由Rno唯一标识，表I由Sno唯一标识，表P由Pno唯一标识。

《数据库系统原理》
厦门大学计算机科学与技术系
林子雨
ziyulin@xmu.edu.cn



课堂作业

- 题目2：编写一个存储过程，输入“鞋子品牌”和“预计销售量”（鞋子的预计销售量），调整该品牌下的每一款鞋子的进货量，对进货表P增加对应鞋子的进货记录。如果该品牌下的一款鞋子的库存量小于等于输入的“预计销售量”且两者差值大于100，进货数量为预计销售量；如果该款鞋子的库存量小于输入的“预计销售量”且两者差值小于等于100，则进货数量为预计销售量的50%；如果该鞋子的库存量大于该输入的“预计销售量”，则进货数量为预计销售量的10%，然后打印出“鞋子编号-预计销售量-进货量”。（进货记录中进货编号为自动增长标识类型，进货单价统一为299，进货时间为当天日期）


```
create pro adjustnum
@input_Sbrand varchar(20),
@input_Count int
as
begin
    declare @Sno int;
    declare @Inum int;
    declare @new_num int;
    declare cur cursor for
    select S.Sno,Inum from S,I
    where S.Sbrand=@input_Sbrand and I.Sno=S.Sno;
    begin
        if (@input_Count-@Inum>=100)
        begin
            set @new_num=@input_Count;
        end
        else if (@input_Count-@Inum>=0)
        begin
            set @new_num=@input_Count*0.5;
        end
        else if (@input_Count-@Inum<0)
        begin
            set @new_num=@input_Count*0.1;
        end
        insert into P values(@Sno,@new_num,299,'2020-5-15')
        print str(@Sno)+'-'+str(@input_Count)+'-'+str(@new_num);
        fetch next from cur into @Sno,@Inum;
    end
    close cur
    deallocate cur
end
```