实验二

091220106
王晓东
基地班

一、 表格描述

本次实验创建 5 张表，描述如下：

1. 职工（姓名，工号，出生日期，家庭地址，年薪，管理员工号，所在部门编号）；
2. 部门（部门名称，部门编号，部门负责人的工号）；
3. 项目（项目名称，项目编号，所在城市，主管部门编号）；
4. 工作（职工工号，项目编号，工作时间）；
5. 家属（职工工号，家属的姓名，家属的性别）。

二、 表格名字、属性名取值类型的设计

按照要求，设计的表名和属性名全为英文名，如下：

1. Staff(SNAME char(20),SNO int,BIRTHDAY char(8),ADDRESS char(50),SALARY double,MNO int,DNO int);
2. Dept(DNAME char(20),DNO int,MNO int);
3. Project(PNAME char(50),PNO int,CITY char(20),DNO int);
4. JOB(SNO int,PNO int,TIME int);
5. FAMILIES(SNO int,FMNAME char(20),SEX char(6));

三、 关系模式中的数据约束

我们将需要进行的完整性数据约束分为以下三条：***属性级约束***、***元组级约束***以及***全局约束***。

a) **属性级约束**（域约束）

数据类型的约束已经在上面给出，下面给出属性的取值范围以及非空约束。

i. **Staff** 中的 SNAME、SNO、BIRTHDAY、ADDRESS、SALARY 为非空。公司中的 BOSS 可以没有 MNO 以及 DNO，所以这两项可以为空。

ii. **Dept** 中的 DNAME 和 DNO 为非空，MNO 可以为空，因为在下任主管上任之前，一个部门可以暂时没有主管，可以设置为 null。

iii. **Project** 中的属性均设置为非空。

iv. **JOB** 中的属性均设置为非空，JOB 中的 **TIME** 定义为该项目和员工签订的合同的时长。

v. **FAMILIES** 中的属性均为非空，而且 **SEX** 只能为“**female**”或“**male**”之一。

b) **元组级约束**（表约束）

这里主要给出主键候选键和外键的约束，以及属性间的约束。

i. **Staff** 中的 **SNO** 为主键；**MNO** 为外键，引用自 Staff 的 SNO，对于 UPDATE 为 Restrict，对于 DELETE 为 set null，因为当某个主管人员调离职位时，在下任主管上任之前，可以暂时没有主管，而设置为 null；**DNO** 为外键，引用自 Dept 的 DNO，对于 UPDATE 为 Restrict，对于 DELETE 为 set null，因为当某个部门解散时，职员在转到其它部门前可以有一段时间没有所属部门。

ii. **Dept** 中的 **DNO** 为主键；**MNO** 为外键，引用自 Staff 的 SNO，对于 UPDATE 为 Restrict，对于 DELETE 为 set null，因为当某个主管人员调离职位时，在下任主管上任之前，可以暂时没有主管，而设置为 null。

iii. **Project** 中的 **PNO** 为主键；**DNO** 为外键，引用自 Dept 中的 DNO，对于 UPDATE 为 Restrict，对于 DELETE 为 Casecade，因为当这个部门被解散时，这个项目理应被终止。

iv. **JOB** 中的**{SNO，PNO}**为主键；**SNO** 为外键，引用自 Staff 中的 SNO，**PNO** 为外键，引用自 Project 的 PNO，它们对于 UPDATE 均为 Restrict，对于 DELETE 均为 Casecade，应为对应的职员离职或者对应的项目终止，那这个工作肯定是无效的。

v. **FAMILIES** 中的**{SNO，FMNAME}**为主键，**SNO** 为外键，引用自 Staff 中的 SNO，对于 UPDATE 为 Restrict，对于 DELETE 为 Casecade，因为当员工离职时，这项记录也就无效了。

c) **全局约束**（断言 assertion）

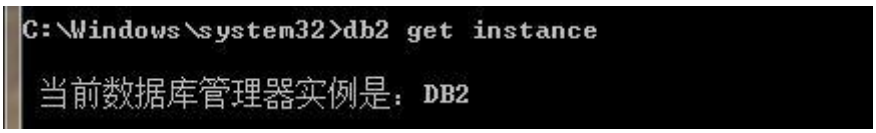
还有一种数据约束是全局约束，但在这道题中似乎为体现出来，也不便做相应的分析。

四、 创建数据库

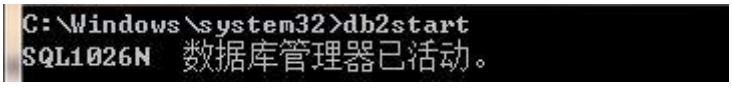
在上面的分析完成后，我们开始在 DB2 上进行实验，实现上面的设计。为了要创建属于自己的表，我们需要创建一个数据库，而为了创建数据库，我们需要创建一个实例（instance）。

由于创建新的实例需要大量的硬件资源，所以，这里使用已有的实例来创建数据库，而不再创建新的实例。

安装完 IBMDB2 之后，我们以管理者权限运行 db2cmd，键入 db2 get instance 的命令，能够查看当前已有的实例，DB2 的安装默认安装好了一个实例 DB2，如下图所示：



为了在这个实例下创建新的数据库，我们先启动数据库管理器，如下：



然后在当前的实例中创建下面的数据库，命名为“MyDB1”，如下：

数据库创建成功，我们可以在控制中心相应的实例下见到刚创建的数据库“MYDB1”，如下：

之后就可以开始对数据库进行相应的操作了。

五、 表格创建（含数据约束）

在进行相应的操作前，我们先要连接数据库，本次连接使用默认的用户名，即当前系统用户名，所以不用指出用户名和相应的密码，如下：

```
connect to MYDB1
```

```
---------------------------- 输入的命令 ----------------------------
connect to MYDB1;
------------------------------------------------------------------------
connect to MYDB1

  数据库连接信息

数据库服务器        = DB2/NT 9.7.4
SQL 授权标识        = MOMO
本地数据库别名      = MYDB1


与目标的 JDBC 连接已成功。
```

然后，为了方便管理统一管理即将创建的几张表，下面创建一个模式DBEXP2，使用 create schema 名命令。

```
create schema DBEXP2;
```

```
---------------------------- 输入的命令 ----------
create schema DBEXP2;
--------------------------------------------------
create schema DBEXP2
DB20000I  SQL 命令成功完成。
```

下面开始创建 5 张表。注意，下面的截图的源代码均能在文件 **Company.sql** 中找到。

1. 创建员工表（Staff）

```
create table Staff (
    SNAME char(20) not null,
    SNO int not null,
    BIRTHDAY char(8) not null,
    ADDRESS char(50) not null,
    SALARY double not null,
    MNO int,
    DNO int,                          --BOSS的MNO和DNO可以没有value
    primary key (SNO),                --set SNO as primary key

    constraint FK_STAFF_MNO foreign key (MNO) references Staff(SNO)
        on UPDATE restrict on DELETE set null
        --管理人员本身也是员工，所以要设置到本身的外键
        --on DELETE 设置为set null，因为当某个主管人员调离职位时，
        ---在下任主管上任之前，可以设置为null，可以没有管理人员
    );
```

第一次定义时，未含有 Dept 的 DNO 外键的约束，因为此时还没有 Dept 表的定义。

2. 创建部门表（Dept）

```
create table Dept (
    DNAME char(20) not null,
    DNO int not null,
    MNO int,                          --在下任主管上任之前，可以设置为null
    primary key (DNO),                --set DNO as primary key
    constraint FK_DEPT_MNO foreign key(MNO) references Staff(SNO)
        on UPDATE restrict on DELETE set null
        --on DELETE设置为set null，因为当某个主管人员调离职位时，
        --在下任主管上任之前，可以设置为null
    );
```

部门表定义的到 Staff 的外键。

3. 在表 Staff 中添加到 Dept 的外键

```
--此时部门表已经创建完成，在Staff中，可以创建Dept的外键，
---下面的语句就是在Staff中增加外键约束
alter table Staff
    add constraint FK_STAFF_DNO foreign key (DNO) references Dept(DNO)
        on UPDATE restrict on DELETE set null;
        --设置Staff表的外键DNO，引用表Dept的DNO
        --on DELETE 设置为set null，因为当某个部门解散时，
        ---职员在转到其它部门前可以有一段时间没有所属部门
--此时Staff已经完全创建好了
```

在上面的 Dept 定义好后可定义 Staff 中的外键 DNO。关于该外键的设置方式已经在上个环节中说明，此处不再赘述。

4. 创建项目表（Project）

```
create table Project(
    PNAME char(50) not null,
    PNO int not null,
    CITY char(20) not null,
    DNO int not null,
    primary key (PNO),                --set PNO as primary key

    constraint FK_PROJECT_DNO foreign key (DNO) references Dept(DNO)
        on UPDATE restrict on DELETE Cascade
        --设置到表Dept的外键DNO。
        --on DELETE 设置为cascade，因为当某个部门解散时，
        ---这个项目也就随之放弃了，可以随之删去
    );
```

5. 创建工作表（JOB）

```
create table JOB(
    SNO int not null,
    PNO int not null,
    TIME int not null,      --这份合同的时长，单位是day
    primary key (SNO,PNO),      --set {SNO,PNO} as primary key

    constraint FK_JOB_SNO foreign key (SNO) references Staff(SNO)
        on UPDATE restrict on DELETE Cascade,
        --有工作的人本身就是员工，所以设置到Staff的外键。
        --on DELETE 设置为Cascade，因为放该员工离职时，
        ---其工作也被撤销了，需要被删除

    constraint FK_JOB_PNO foreign key (PNO) references Project(PNO)
        on UPDATE restrict on DELETE Cascade
        --所有工作均是一个项目，所以要设置到Project的外键
        --on DELETE 设置为Cascade，因为当某个项目完成或者终止时，该项JOB也应该被注销
    );
```

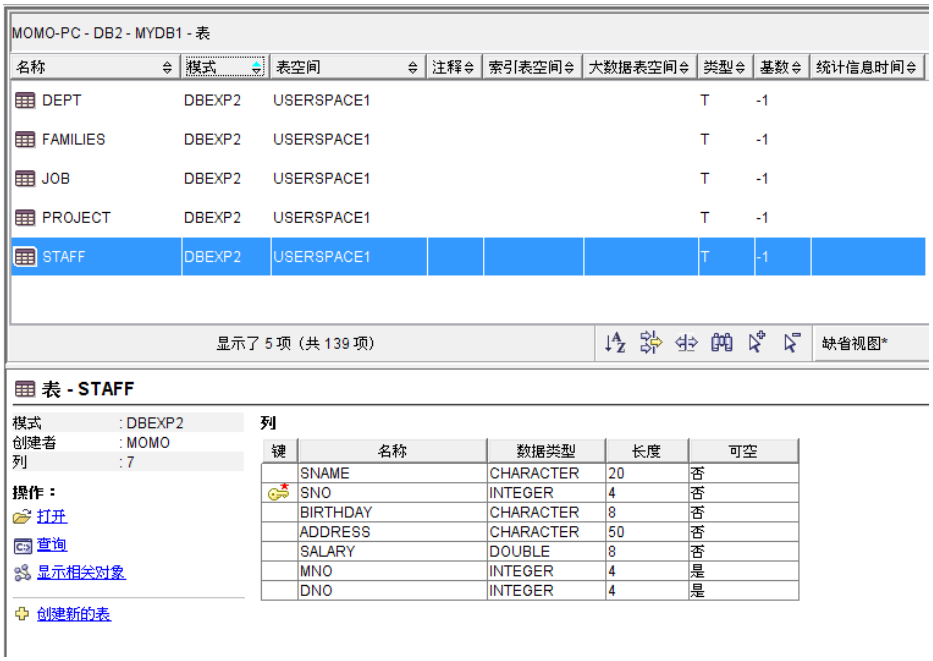
6. 创建家属表（FAMILIES）

```
create table FAMILIES(
    SNO int not null,
    FMNAME char(20) not null,
    SEX char(6) not null,
    primary key(SNO,FMNAME),        --set {SNO,FMNAME} as primary key
        --不是将SNO作为主键，因为可能一个员工会注册多个家属
        --不是将FMNAME作为主键，因为可能有不同的员工有家属同名

    constraint SEX_VALUE check (SEX = 'female' or SEX = 'male'),
        --创建性别取值约束，SEX的取值只能为female或者male两种。

    constraint FK_FAMILIES_SNO foreign key (SNO) references Staff(SNO)
        on UPDATE restrict on DELETE Cascade
        --on DELETE 设置为Cascade，因为放该员工离职时，
        ---其家属记录也应该被删去
    );
```

经过上面的步骤，创建完成了 5 张表，我们可以在控制中心看到它们（已经设置模式为 DBEXP2 的过滤）：



六、 数据添加

下面，对 5 张表添加数据，数据的添加也可以在 **Company.sql** 中找到。
这里为了符合实际世界的规则，采用下面的先后方式，添加具体数据

1. 添加部门数据，此时还没有实际员工，所以主管的那项全部为 null

```
--此时还没有实际员工，所以主管的那项全部为null
INSERT INTO Dept VALUES
--(DNAME,           DNO,MNO)
('Design',          10,null),              --Department of Design
('Marketing',       20,null),              --Department of Marketing
('HumanResource',   30,null)               --Department of HR
```

2. 添加员工数据，此时可以规定某些员工为某个部门的主管

```
--添加员工数据，此时可以规定某些员工为某个部门的主管
INSERT INTO Staff VALUES
--(SNMAE,   SNO, BIRTHDAY,  ADDRESS,           SOLARY,MNO,DNO)
('Smith',   1000,'19680608','GreatStreet 106', 30000,null,null),       --Boss
('Bob',     1001,'19781018','LongWall 23',     20000,1000,10),     --Dept 10 Manager
('Mark',    1002,'19721118','ManCack 21',      20000,1000,20),     --Dept 20 Manager
('Green',   1003,'19700327','LongWall 40',     20000,1000,30),     --Dept 30 Manager
('Yorky',   1004,'19700227','ShortMell 90',    10000,1001,10),     --Dept 10 Staff
('Milk',    1005,'19710227','SoMell 102',      10000,1001,10),     --Dept 10 Staff
('Jam',     1006,'19800310','TomHall 34',      10000,1002,20),     --Dept 20 Staff
('Nancy',   1007,'19790212','LintonPark 73',   10000,1002,20),     --Dept 20 Staff
('Wallet',  1008,'19800513','MonkeyD 24',      10000,1003,30),     --Dept 30 Staff
('Tylan',   1009,'19790809','TunDa 79',        10000,1003,30)      --Dept 30 Staff
```

3. 更新部门的数据表，添加具体的主管工号

```
--更新部门的数据表，添加具体的主管工号
UPDATE Dept SET MNO=1001 WHERE DNO=10;
UPDATE Dept SET MNO=1002 WHERE DNO=20;
UPDATE Dept SET MNO=1003 WHERE DNO=30;
```

4. 添加项目表数据

```
--添加项目表数据
INSERT INTO Project VALUES
--(PNAME,           PNO,CITY,       DNO)
('Project Lumia',   100,'NewYork',  10),
('Project Kindom',  200,'Beijing',  10),
('Project Monkey',  300,'Tokyo',    20),
('Project Banana',  400,'ShenZhen', 30)
```

5. 添加工作表数据

```
--添加工作表数据
INSERT INTO JOB VALUES
--(SNO, PNO,    TIME)
(1001,  100,    30),
(1004,  100,    30),
(1001,  200,    60),
(1005,  200,    60),
(1002,  300,    60),
(1006,  300,    60),
(1007,  300,    60),
(1003,  400,    65),
(1008,  400,    65),
(1009,  400,    65)
```

6. 添加家属表数据

```
--添加家属表数据
INSERT INTO FAMILIES VALUES
--(SNO,FMNAME,   SEX)
(1000,'Merry',  'female'),
(1001,'Tom',    'male'),
(1001,'Lily',   'female'),
(1002,'HanMei', 'female'),
(1007,'Pom',    'male')
```

经过上面的六个步骤，数据已经添加完全，可以在控制中心进行查看，如下面的 Staff 表：

SNAME	SNO	BIRTHDAY	ADDRESS	SALARY	MNO	DNO
Smith	1000	19680608	GreatStreet 106 ...	30,000		
Bob	1001	19781018	LongWall 23 ...	20,000	1000	10
Mark	1002	19721118	ManCack 21 ...	20,000	1000	20
Green	1003	19700327	LongWall 40 ...	20,000	1000	30
Yorky	1004	19700227	ShortMell 90 ...	10,000	1001	10
Milk	1005	19710227	SoMell 102 ...	10,000	1001	10
Jam	1006	19800310	TomHall 34 ...	10,000	1002	20
Nancy	1007	19790212	LintonPark 73 ...	10,000	1002	20
Wallet	1008	19800513	MonkeyD 24 ...	10,000	1003	30
Tylan	1009	19790809	TunDa 79 ...	10,000	1003	30

我们可以看到 Boss Smith 的所属部门号以及主管人员均为 null。

七、 进行影响数据约束的操作

下面，为了观察数据约束设置对数据操作的影响，我们进行几项操作来进行观察。

case 1. 插入数据的主键的值为空或者重复

```
INSERT INTO Staff VALUES
    ('Tylan',   null,'19790809','TunDa 79',         10000,1003,30)       --主键为null
```

使用上面的语句，会得到报错，提示如下：

```
SQL 处理期间，它返回：
SQL0407N  不允许对 NOT NULL 列 "TBSPACEID=2, TABLEID=4, COLNO=1" 赋予空值。
SQLSTATE=23502
```

因为是主键，所以一定是 not null 的。

```
INSERT INTO Staff VALUES
    ('abc', 1009,'19790809','TunDa 79',         10000,1003,30)       --主键重复
```

使用上面的语句，得到出错提示如下：

```
SQL 处理期间，它返回：
SQL0803N  INSERT 语句、UPDATE 语句或由 DELETE
语句导致的外键更新中的一个或多个值无效，因为由 "1"
标识的主键、唯一约束或者唯一索引将表 "DBEXP2.STAFF"
的索引键限制为不能具有重复值。  SQLSTATE=23505
```

这体现了主键取值的不可重复性。

case 2. 插入数据破坏属性值的取值范围

在表 Families 中有一项是 SEX，该项有一个取值约束，SEX 只能是 femalehu 或者是 male。使用下面的语句会出错：

```
INSERT INTO FAMILIES VALUES
    (1009,'Mary',   'Women')        --在SEX_VALUE的约束下,SEX的取值只能是female或者male
```

出错提示为：

```
SQL 处理期间，它返回：
SQL0545N  因为行不满足检查约束
"DBEXP2.FAMILIES.SEX_VALUE"，所以不允许所请求的操作。  SQLSTATE=23513
```

因为插入的值不满足检查约束 SEX_VALUE。

case 3. 外键设置时，DELETE 的方式为 Cascade

```
DELETE FROM Staff Where SNO = 1007;
--此时，可以看到在FAMILIES，JOB中含有SNO=1007的项同时被删除
```

此时，可以看到在 FAMILIES，JOB 中含有 SNO=1007 的项均被删除，如下：

SNO	PNO	TIME
1001	100	30
1004	100	30
1001	200	60
1005	200	60
1002	300	60
1006	300	60
1003	400	65
1008	400	65
1009	400	65

SNO	FMNAME	SEX
1000	Merry	female
1001	Tom	male
1001	Lily	female
1002	HanMei	female

其中已经没有 1007 的那个 JOB 以及 Families 记录。

case 4. 外键设置时，DELETE 的方式为 set null

假设工号为 1003 的职员被撤职（1003 是 Dept30 的主管）

那么执行下面的命令：

```
--假设工号为1003的职员被撤职（1003是Dept30的主管）
DELETE FROM Staff Where SNO = 1003;
```

此时，可以看到在 Staff 中，原先由 1003 主管的 1008，1009 的管理人员的工号为 null，另外 Dept 中 Dept30 的主管也变为为 null（等待新的主管接替），如下：

Wallet	1008	19800513	MonkeyD 24	...	10,000	30
Tylan	1009	19790809	TunDa 79	...	10,000	30

Marketing	20	1002
HumanResourc...	30	

case 5. 外键设置时，UPDATE 的方式为 Restrict

此时，假设上级命令，将 Dept10 的部门号改为 40，那么执行下面的命令：

```
--假设上级命令，将Dept10的部门号改为40
UPDATE Dept SET DNO=40 WHERE DNO=10;
```

但是，实际上不能运行，由于 Staff、Project 的外键设置的 UPDATE 均为

Restrict，所以这项更新会报错。

```
SQL 处理期间，它返回：
SQL0531N  不能更新关系 "DBEXP2.STAFF.FK_STAFF_DNO" 的父行中的父键。
SQLSTATE=23001
```

为了防止出现逻辑错误的情况（如果上面的操作成功，所有属于部门 10 的员工就会无辜的突然失业...），这样做，是有必要的。

八、 创建触发器

在这个环节，将将建一个触发器 inc_salary，当职工参加一个新的项目时，年薪增加 2%。

```
--创建一个触发器，当职工参加一个新的项目时，年薪增加2%
create trigger inc_salary
    after insert on JOB
    referencing NEW as N
    for each row mode DB2SQL
        update Staff
        set Staff.SALARY = Staff.SALARY * 1.02
        where Staff.SNO = N.SNO;
```

下面我们来观察这个触发器的行为。

首先我们给出此时 Staff 表：

SNAME	SNO	BIRTHDAY	ADDRESS	SALARY	MNO	DNO
Smith	1000	19680608	GreatStreet 106 ...	30,000		
Bob	1001	19781018	LongWall 23 ...	20,000	1000	10
Mark	1002	19721118	ManCack 21 ...	20,000	1000	20
Yorky	1004	19700227	ShortMell 90 ...	10,000	1001	10
Milk	1005	19710227	SoMell 102 ...	10,000	1001	10
Jam	1006	19800310	TomHall 34 ...	10,000	1002	20
Wallet	1008	19800513	MonkeyD 24 ...	10,000		30
Tylan	1009	19790809	TunDa 79 ...	10,000		30

请注意上面 1004 的 SALARY 值，为 10000。下面我们给 1004 增加一个 JOB，对应的项目为 200：

```
--给1004增加一个JOB，对应的项目为200
INSERT INTO JOB VALUES (1004,200,30)
```

在观察 Staff 表如下：

SNAME	SNO	BIRTHDAY	ADDRESS	SALARY	MNO	DNO
Smith	1000	19680608	GreatStreet 106 ...	30,000		
Bob	1001	19781018	LongWall 23 ...	20,000	1000	10
Mark	1002	19721118	ManCack 21 ...	20,000	1000	20
Yorky	1004	19700227	ShortMell 90 ...	10,200	1001	10
Milk	1005	19710227	SoMell 102 ...	10,000	1001	10
Jam	1006	19800310	TomHall 34 ...	10,000	1002	20
Wallet	1008	19800513	MonkeyD 24 ...	10,000		30
Tylan	1009	19790809	TunDa 79 ...	10,000		30

我们可以看到 1004 的员工的 SALARY 已经变为了 10200。

九、 创建并授权新用户。

在这个环节，我们需要对一个用户进行授权。

为什么不说要创建用户，使用因为，经过查阅资料，发现 DB2 的用户认证是交给操作系统会其他第三方认证系统的，所以 DB2 的用户创建，本质上与 DB2 没有关系，换句话说，即使你使用平常在自己系统中创建的用户，你也可以在 DB2 中使用，所以本次本不再仔细分析如何创建用户（只要在操作系统中创建即可），这里使用的另一个用户是在安装 DB2 过程中默认创建的 db2admin 来进行演示。

首先，我们可以来看一下未经授权的 db2admin 能否访问之前用户 MOMO 新建的表。

我们先断开先前的连接，使用新的用户 db2admin 登录，再进行访问操作。

```
disconnect current;
connect to MYDB1 user db2admin using ********;
select * from "DBEXP2".JOB;
```

结果是："DB2ADMIN" 不具有对对象 "DBEXP2.JOB" 执行操作 "SELECT"的必需权限或特权。如下图：

```
select * from "DBEXP2"JOB
SQL0551N "DB2ADMIN" 不具有对对象 "DBEXP2.JOB" 执行操作 "SELECT"
的必需权限或特权。 SQLSTATE=42501
```

提示没有访问权限。

我们再切回之前默认的连接（MOMO 是我的默认用户名...）。

```
--重新以MOMO的用户名连接，进行相应的授权
disconnect current;
connect to MYDB1;
grant SELECT on "DBEXP2".JOB
    to db2admin with grant option
```

将对模式 DBEXP2 下的 JOB 表的 SELECT 权限交给 db2admin，并具有授权权限。

然后再切回 db2admin 下，进行同样的操作：

```
--重新使用db2admin进行连接，在进行同样的查询
disconnect current;
connect to MYDB1 user db2admin using ********;
select * from "DBEXP2".JOB;
--得到正确的结果
```

此时能够查询表 JOB：

```
select * from "DBEXP2".JOB

SNO         PNO         TIME
----------- ----------- -----------
      1001         100          30
      1004         100          30
      1001         200          60
      1005         200          60
      1002         300          60
      1006         300          60
      1004         200          30
      1008         400          65
      1009         400          65

 9 条记录已选择。
```