

# 数据库系统上机报告

西安电子科技大学

计算机科学与技术学院

1603019 班

张俊华

16030199025

2018 年 12 月



# 上机内容

---

## 实验环境

---

- SQLServer 2017
- 客户端：
  - Navicat Premium 12.1.10 64-bit
  - Visual Studio Code 1.29.1

## 一、需求分析

---

用DDL语句创建习题中的SPJ数据库及基本表、约束、视图等数据库对象；用DML语句插入、删除、修改教材中的实例数据；选择完成作业中的查询（不少于10个）；编写存储过程，实现按供应商号查询该供应商的供应信息；结合教材中SPJ数据库，修改关系S，增加等级属性列（level），编写触发器，当更新SPJ表中的QTY列时，取该供应商的供应量总和除以100作为其对应等级。

## 信息要求

需要为 供应商（S）、零件表（P）、工程项目表（J）建立数据库以存储信息

- 供应商 需要记录：供应商代码（SNO）、供应商姓名（SNAME）、供应商状态（STATUS）、供应商所在城市（CITY）、供应商等级
- 零件信息有：零件代码（PNO）、零件名（PNAME）、颜色（COLOR）、重量（WEIGHT）
- 工程项目信息有：工程项目代码（JNO）、工程项目名（JNAME）、工程项目所在城市（CITY）
- 供应情况有：供应商代码（SNO）、零件代码（PNO）、工程项目代码（JNO）、供应数量（QTY）

## 处理要求

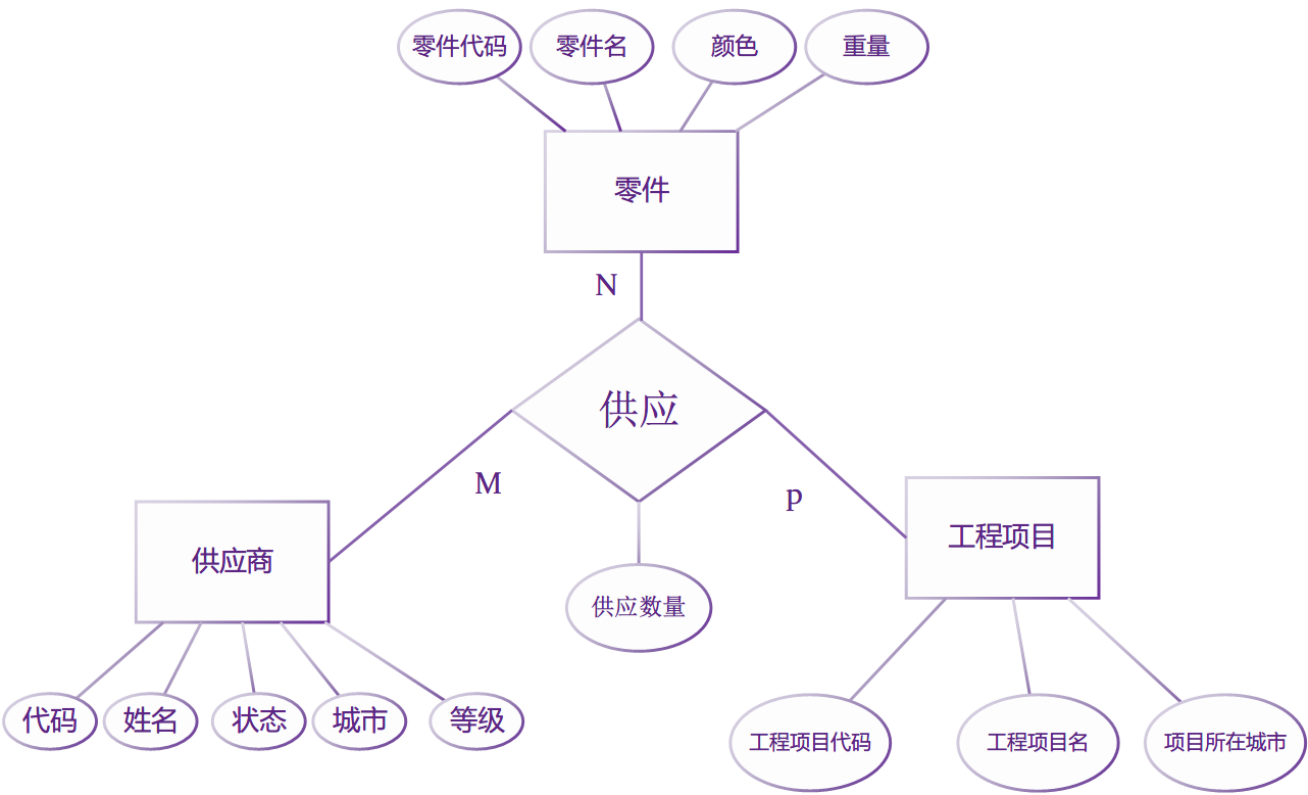
供应商等级需要根据供应商的供应量总和自动计算

## 安全性与完整性要求

- 供应商
  - 供应商代码唯一
  - 供应商代码、供应商姓名、供应商状态、供应商所在城市 非空
- 零件信息
  - 零件代码唯一
  - 零件名、颜色、重量 非空
- 工程项目信息
  - 工程项目代码非空唯一
  - 工程项目名、工程项目所在城市 非空
- 供应情况
  - 供应商代码、零件代码、工程项目代码、供应数量 非空

## 二、概念结构设计

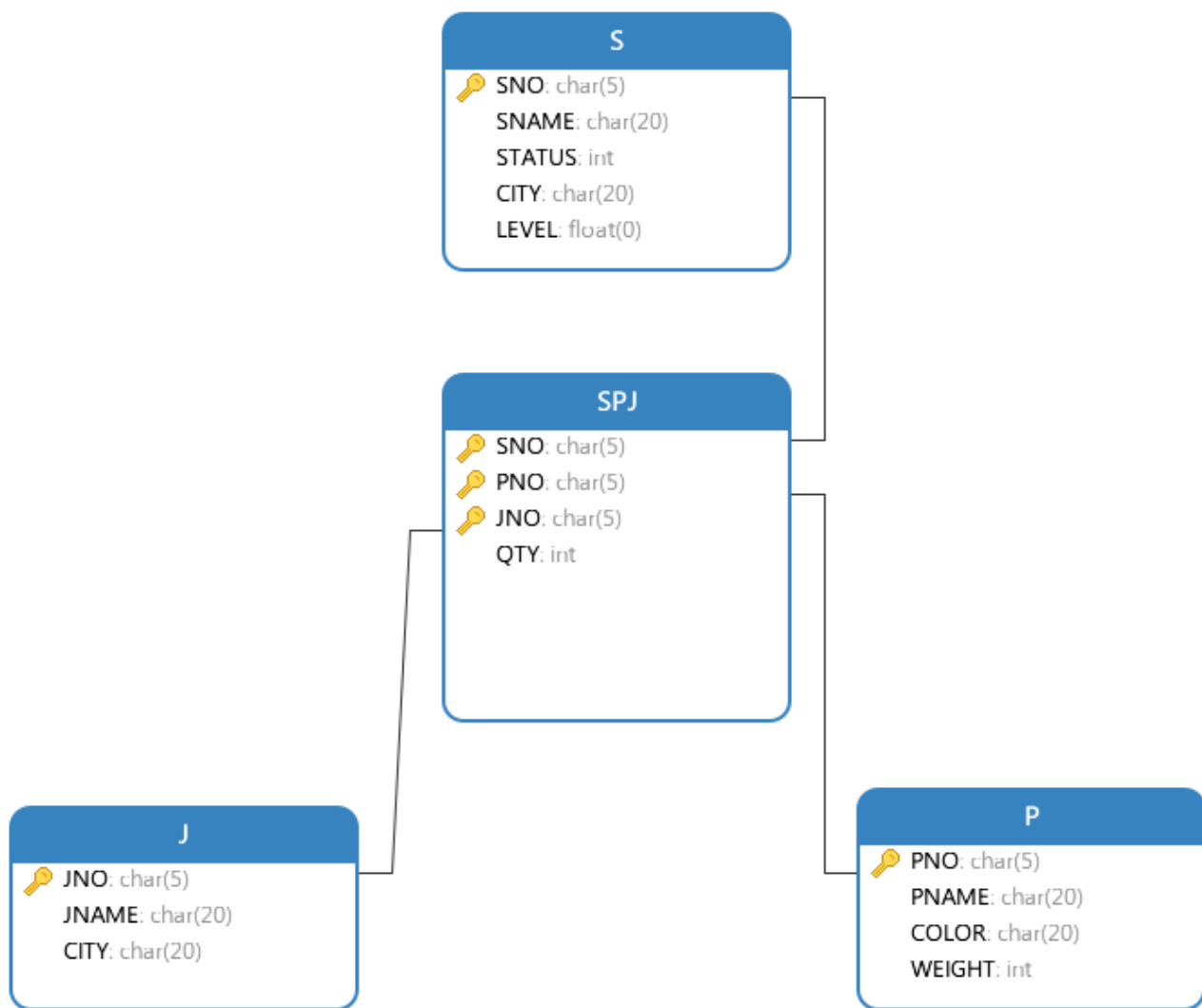
根据需求，设计 E-R图如下：



## 三、逻辑结构设计

根据 E-R 图，设计关系模型如下：

对每个实体定义的属性：



## 四、源代码以及详细实现

### 1.用DDL语句创建习题中的SPJ数据库及基本表、约束、视图等数据库对象；

```
1 CREATE TABLE S(  
2     SNO CHAR(5) PRIMARY KEY,  
3     SNAME CHAR(20),  
4     STATUS INTEGER,  
5     CITY CHAR(20),  
6 );  
7  
8 CREATE TABLE P(  
9     PNO CHAR(5) PRIMARY KEY,  
10    PNAME CHAR(20),  
11    COLOR CHAR(20),  
12    WEIGHT INTEGER,  
13 );  
14  
15 CREATE TABLE J(  
16     JNO CHAR(5) PRIMARY KEY,  
17     JNAME CHAR(20),
```

```

18     CITY CHAR(20),
19 );
20
21 CREATE TABLE SPJ(
22     SNO CHAR(5) NOT NULL,
23     PNO CHAR(5) NOT NULL,
24     JNO CHAR(5) NOT NULL,
25     QTY INTEGER,
26     PRIMARY KEY (SNO,PNO,JNO,QTY),
27     FOREIGN key (PNO) REFERENCES P(PNO),
28     FOREIGN key (JNO) REFERENCES J(JNO),
29     FOREIGN key (SNO) REFERENCES S(SNO),
30 );

```

对象
\* 无标题 - 查询

保存
查询创建工具
美化 SQL
代码段

MS SQLserver
learnsql2
dbo
运行
停止
解释

```

1 CREATE TABLE S(
2     SNO CHAR(5) PRIMARY KEY,
3     SNAME CHAR(20),
4     STATUS INTEGER,
5     CITY CHAR(20),
6 );
7
8 CREATE TABLE P(
9     PNO CHAR(5) PRIMARY KEY,
10    PNAME CHAR(20),
11    COLOR CHAR(20)
12 );
13
14 CREATE TABLE J(
15     JNO CHAR(5) PRIMARY KEY,
16     JNAME CHAR(20),
17     CITY CHAR(20),
18 );
19
20 CREATE TABLE SPJ(
21     SNO CHAR(5) NOT NULL,
22     PNO CHAR(5) NOT NULL,
23     JNO CHAR(5) NOT NULL,
24     QTY INTEGER,
25     PRIMARY KEY (SNO,PNO,JNO,QTY),
26     FOREIGN key (PNO) REFERENCES P(PNO),
27     FOREIGN key (JNO) REFERENCES J(JNO),
28     FOREIGN key (SNO) REFERENCES S(SNO),
29 );

```

信息

```

> OK
> 时间: 0.004s

CREATE TABLE J(
    JNO CHAR(5) PRIMARY KEY,
    JNAME CHAR(20),
    CITY CHAR(20),
);
> OK
> 时间: 0.003s

CREATE TABLE SPJ(
    SNO CHAR(5) NOT NULL,
    PNO CHAR(5) NOT NULL,
    JNO CHAR(5) NOT NULL,
    QTY INTEGER,
    PRIMARY KEY (SNO,PNO,JNO,QTY),
    FOREIGN key (PNO) REFERENCES P(PNO),
    FOREIGN key (JNO) REFERENCES J(JNO),
    FOREIGN key (SNO) REFERENCES S(SNO),
);
> OK
> 时间: 0.006s

```

建立 `spj_detail` 视图, 实现 供应商供应情况详细信息的查询。

```

1 CREATE VIEW spj_detail
2 AS
3 SELECT J.JNO,J.JNAME,J.CITY as
   JCITY,P.PNO,P.PNAME,P.COLOR,P.WEIGHT,S.SNO,S.SNAME,S.CITY AS SCITY
4 FROM J,P,S,SPJ
5 WHERE J.JNO = SPJ.JNO and
6        P.PNO = SPJ.PNO and
7        S.SNO = SPJ.SNO
8 GO

```

执行结果：

## 信息

```

CREATE VIEW spj_detail
AS
SELECT J.JNO,J.JNAME,J.CITY as JCITY,P.PNO,P.PNAME,P.COLOR,P.WEIGHT,S.SNO,S.SNAME,S.CITY AS SCITY
      FROM J,P,S,SPJ
      WHERE J.JNO = SPJ.JNO and
            P.PNO = SPJ.PNO and
            S.SNO = S.SNO
> OK
> 时间：0.028s

```

JNO	JNAME	JCITY	PNO	PNAME	COLOR	WEIGHT	SNO	SNAME	SCITY
J1	三建	北京	P1	螺母	红	12	S1	精益	天津
J1	三建	北京	P1	螺母	红	12	S2	盛锡	北京
J1	三建	北京	P1	螺母	红	12	S3	东方红	北京
J1	三建	北京	P1	螺母	红	12	S4	丰泰盛	天津
J1	三建	北京	P1	螺母	红	12	S5	为民	上海
J3	弹簧厂	天津	P1	螺母	红	12	S1	精益	天津
J3	弹簧厂	天津	P1	螺母	红	12	S2	盛锡	北京
J3	弹簧厂	天津	P1	螺母	红	12	S3	东方红	北京
J3	弹簧厂	天津	P1	螺母	红	12	S4	丰泰盛	天津
J3	弹簧厂	天津	P1	螺母	红	12	S5	为民	上海
J4	造船厂	天津	P1	螺母	红	12	S1	精益	天津
J4	造船厂	天津	P1	螺母	红	12	S2	盛锡	北京
J4	造船厂	天津	P1	螺母	红	12	S3	东方红	北京
J4	造船厂	天津	P1	螺母	红	12	S4	丰泰盛	天津
J4	造船厂	天津	P1	螺母	红	12	S5	为民	上海
J2	一汽	长春	P2	螺栓	绿	17	S1	精益	天津
J2	一汽	长春	P2	螺栓	绿	17	S2	盛锡	北京
J2	一汽	长春	P2	螺栓	绿	17	S3	东方红	北京
J2	一汽	长春	P2	螺栓	绿	17	S4	丰泰盛	天津
J2	一汽	长春	P2	螺栓	绿	17	S5	为民	上海
J1	三建	北京	P3	螺丝刀	蓝	14	S1	精益	天津
J1	三建	北京	P3	螺丝刀	蓝	14	S2	盛锡	北京
J1	三建	北京	P3	螺丝刀	蓝	14	S3	东方红	北京
J1	三建	北京	P3	螺丝刀	蓝	14	S4	丰泰盛	天津
J1	三建	北京	P3	螺丝刀	蓝	14	S5	为民	上海

## 2. 用DML语句插入、删除、修改教材中的实例数据；

```

1 INSERT
2 into [dbo].[S]
3 VALUES
4 ('S1','精益',20,'天津'),
5 ('S2','盛锡',10,'北京'),
6 ('S3','东方红',30,'北京'),
7 ('S4','丰泰盛',20,'天津'),
8 ('S5','为民',30,'上海');
9

```

```

10 INSERT
11 into [dbo].[P]
12 VALUES
13 ('P1','螺母','红',12),
14 ('P2','螺栓','绿',17),
15 ('P3','螺丝刀','蓝',14),
16 ('P4','螺丝刀','红',14),
17 ('P5','凸轮','蓝',40),
18 ('P6','齿轮','红',30);
19
20 INSERT
21 into [dbo].[J]
22 VALUES
23 ('J1','三建','北京'),
24 ('J2','一汽','长春'),
25 ('J3','弹簧厂','天津'),
26 ('J4','造船厂','天津'),
27 ('J5','机车厂','唐山'),
28 ('J6','无线电厂','常州'),
29 ('J7','半导体厂','南京');
30
31 INSERT
32 into SPJ
33 VALUES
34 ('S1','P1','J1',200),
35 ('S1','P1','J3',100),
36 ('S1','P1','J4',700),
37 ('S1','P2','J2',100),
38 ('S2','P3','J1',400),
39 ('S2','P3','J2',200),
40 ('S2','P3','J4',500),
41 ('S2','P3','J5',400),
42 ('S2','P5','J1',400),
43 ('S2','P5','J2',100),
44 ('S3','P1','J1',200),
45 ('S3','P3','J1',200),
46 ('S4','P5','J1',100),
47 ('S4','P6','J3',300),
48 ('S4','P6','J4',200),
49 ('S5','P2','J4',100),
50 ('S5','P3','J1',200),
51 ('S5','P6','J2',200),
52 ('S5','P6','J4',500);

```

实例数据插入后的数据表：

对象	J @learnsql.dbo (MS SQLser...	P @learnsql.dbo (MS SQLse...
开始事务	文本	筛选
JNO	JNAME	CITY
J1	三建	北京
J2	一汽	长春
J3	弹簧厂	天津
J4	造船厂	天津
J5	机车厂	唐山
J6	无线电厂	常州
J7	半导体厂	南京

对象	J @learnsql.dbo (MS SQLser...	P @learnsql.dbo (MS SQLse...
开始事务	文本	筛选
SNO	SNAME	STATUS
S1	精益	20 天津
S2	盛锡	10 北京
S3	东方红	30 北京
S4	丰泰盛	20 天津
S5	为民	30 上海

对象	J @learnsql.dbo (MS SQLser...	P @learnsql.dbo (MS SQLse...
开始事务	文本	筛选
SNO	PNO	JNO
S1	P1	J1
S1	P1	J3
S1	P1	J4
S1	P2	J2
S2	P3	J1
S2	P3	J2
S2	P3	J4
S2	P3	J5
S2	P5	J1
S2	P5	J2
S3	P1	J1
S3	P3	J1
S4	P5	J1
S4	P6	J3
S4	P6	J4
S5	P2	J4
S5	P3	J1
S5	P6	J2
S5	P6	J4

### 3.编写存储过程，实现按供应商号查询该供应商的供应信息

编写存储过程 `get_s_info`，其参数：`@sno CHAR(5)` 为待查询的供应商 `SNO`

```
1 ALTER PROC get_s_info
2 @sno CHAR(5)
3 AS
4 BEGIN
5 SELECT DISTINCT * FROM spj_detail WHERE Sno = @sno
6 END
```

执行存储过程

```
1 exec get_s_info 'S1'
```

执行存储过程结果如下，可以看到，存储过程成功返回了所查询的信息

定义	注释	信息	结果 1	SQL 预览						
JNO	JNAME	JCITY	PNO	PNAME	COLOR	WEIGHT	SNO	SNAME	SCITY	
J1	三建	北京	P1	螺母	红	12	S1	精益	天津	
J3	弹簧厂	天津	P1	螺母	红	12	S1	精益	天津	
J4	造船厂	天津	P1	螺母	红	12	S1	精益	天津	
J2	一汽	长春	P2	螺栓	绿	17	S1	精益	天津	

### 4. 结合教材中SPJ数据库，修改关系S，增加等级属性列（level），编写触发器，当更新SPJ表中的QTY列时，取该供应商的供应量总和除以100作为其对应等级。

信息要求




结合教材中SPJ数据库，修改关系S，增加等级属性列（level），编写触发器，当更新SPJ表中的QTY列时，取该供应商的供应量总和除以100作为其对应等级。

编写触发器 `make_level` 如下：

```
1  ---- 增加等级属性列
2  ALTER TABLE S
3      ADD level FLOAT NULL
4  GO
5
6
7  CREATE TRIGGER make_level
8  ON SPJ
9  AFTER UPDATE,INSERT,DELETE
10 AS
11 BEGIN
12     declare @sumqty int
13     -- select rows from a Table 'spj'
14     SELECT @sumqty = SUM(SPJ.QTY) FROM SPJ,inserted
15     WHERE SPJ.SNO = inserted.SNO
16     UPDATE S set level = @sumqty/100 FROM S, inserted
17     WHERE S.SNO = inserted.SNO
18 END
19 GO
```

执行sql 语句后，可以看到数据库中增加了 `make_level` 触发器：



The screenshot displays the SQL Server Enterprise Manager interface. The 'SPJ' table is selected, and the 'make\_level' trigger is visible in the 'Triggers' tab. The trigger is configured to fire 'After' updates, inserts, and deletes. The trigger definition is as follows:

```
1 BEGIN
2     declare @sumqty int
3     -- Select rows from a Table or View 'spj'
4     SELECT @sumqty = SUM(SPJ.QTY) FROM SPJ,inserted
5     WHERE SPJ.SNO = inserted.SNO /* add search conditions here */
6     UPDATE S set level = @sumqty/100 FROM S, inserted
7     WHERE S.SNO = inserted.SNO
8 END
```

当修改 spj 表中 QTY列时，S 表的对应 `level` 字段会自动更新：

对象

< 1...

P @learnsql.dbo (...)

S @learnsql.dbo (...)

开始事务

文本

筛选

排序

导入

导出

SNO	SNAME	STATUS	CITY	level
▶ S1	精益	20	天津	12
S2	盛锡	10	北京	20
S3	东方红	30	北京	4
S4	丰泰盛	20	天津	6
S5	为民	30	上海	10

## 5. 选择完成作业中的查询

-- 求供应工程 J1 零件的供应商号码 SNO

```
SELECT DISTINCT SNO FROM SPJ
WHERE JNO = 'J1'
```

	SNO
1	S1
2	S2
3	S3
4	S4
5	S5

-- 求供应工程 J1 零件的供应商号码 SNO

```
SELECT DISTINCT SNO FROM SPJ
WHERE JNO = 'J1' AND
      PNO = 'P1';
```

	SNO
1	S1
2	S3

-- (3) 求供应工程 J1 零件为红色的供应商号码 SNO;

```
SELECT DISTINCT Sno
FROM SPJ
WHERE Jno = 'J1' AND
      EXISTS(
        SELECT Pno
        FROM P
        WHERE Color = '红' AND
              P.Pno = SPJ.Pno
      );
```

	Sno
1	S1
2	S3

-- (4) 求没有使用天津供应商生产的红色零件的工程号 JNO;

```
SELECT DISTINCT Jno
FROM J
WHERE not EXISTS(
  SELECT *
  FROM S,P, SPJ
  WHERE SPJ.JNO = J.JNO AND
        S.Sno = SPJ.Sno AND
        S.city = '天津' AND
        P.Pno = SPJ.Pno AND
        P.color = '红'
);
```

	Jno
1	J2
2	J5
3	J6
4	J7

-- (5) 求至少用了供应商 S1 所供应的全部零件的工程号 JNO。

```
SELECT Jno
FROM SPJ AS spj
EXCEPT
SELECT Jno
FROM SPJ AS spja
WHERE EXISTS(
  SELECT *
  FROM SPJ as spjb
  WHERE spjb.SNO = 'S1' AND
        NOT EXISTS (
          SELECT *
          FROM SPJ as spjc
          WHERE spjc.PNO = spjb.PNO AND
                spjc.JNO = spja.JNO
        )
);
```

	Jno
1	J4

-- (1) 找出所有供应商的姓名和所在城市;

```
SELECT Sname ,city
FROM S;
```

	Sname	city
1	精益	天津
2	盛锡	北京
3	东方红	北京
4	丰泰盛	天津
5	为民	上海

-- (2) 找出所有零件的名称、颜色、重量;

```
SELECT Pname,color,weight
FROM P;
```

	Pname	color	weight
1	螺母	红	12
2	螺栓	绿	17
3	螺丝刀	蓝	14
4	螺丝刀	红	14
5	凸轮	蓝	40
6	齿轮	红	30

-- (3) 找出使用供应商 S1 所供应零件的工程号码;

```
SELECT DISTINCT Pno
FROM SPJ
WHERE Sno = 'S1'
```

	Pno
1	P1
2	P2

-- (4) 找出工程项目 J2 使用的各种零件的名称及其数量;

```
SELECT Pname, QTY
FROM P, SPJ
WHERE Jno = 'J2' AND
      P.PNO = SPJ.PNO
```

	Pname	QTY
1	螺栓	100
2	螺丝刀	200
3	凸轮	100
4	齿轮	200

-- (5) 找出上海厂商供应的所有零件号码

```
SELECT DISTINCT Pno
FROM SPJ, S
WHERE S.city = '上海' AND
      SPJ.Sno = S.Sno;
```

	Pno
1	P2
2	P3
3	P6

-- (6) 找出使用上海产的零件的工程名称;

```
SELECT PNAME
FROM P
WHERE EXISTS(
  select *
  FROM S, SPJ
  WHERE P.PNO = SPJ.PNO AND
        S.city = '上海' AND
        SPJ.Sno = S.Sno
```

);

	PNAME
1	螺栓
2	螺丝刀
3	齿轮

-- (7) 找出没有使用天津产的零件的工程号码;

```
SELECT Pno
FROM P
WHERE not EXISTS(
  select *
  FROM S, SPJ
  WHERE P.PNO = SPJ.PNO AND
        S.city != '天津' AND
        SPJ.Sno = S.Sno
```

);

	Pno
1	P4

-- (8) 把全部红色零件的颜色改成蓝色;

```
UPDATE P
SET color = '蓝'
WHERE P.color = '红';
```

-- (9) 由 S5 供给 J4 的零件 P6 改为由 S3 供应, 请作必要的修改;

```
UPDATE SPJ
SET Sno = 'S3'
where Sno = 'S5' AND
      Jno = 'J4' AND
      Pno = 'P6';
```

-- (10) 从供应商关系中删除 S2 的记录, 并从供应情况关系中删除相应的记录;

```
DELETE FROM SPJ
WHERE SNO = 'S2';
DELETE FROM S
WHERE SNO = 'S2';
```

-- (11) 请将 (S2, J6, P4, 200) 插入供应情况关系。

```
INSERT INTO SPJ
VALUES
('S2', 'J6', 'P4', 200);
```

/\*

请为三建工程项目建立一个供应情况的视图，  
包括供应商代码（SNO）、零件代码（PNO）、供应数量（QTY）。

针对该视图完成下列查询：

\*/

```
CREATE VIEW SJ_SUPPLYINFO
```

```
as
```

```
select SNO, PNO, QTY
```

```
FROM SPJ,J
```

```
WHERE J.JNAME = '三建' and
```

```
SPJ.JNO = J.JNO;
```

```
GO
```

--（1）找出三建工程项目使用的各种零件代码及其数量；

```
SELECT PNO, QTY
```

```
From SJ_SUPPLYINFO;
```

The screenshot shows the SQL Server Enterprise Manager interface. The top pane displays the SQL code for creating the view SJ\_SUPPLYINFO. The bottom pane shows the execution results of the query SELECT PNO, QTY FROM SJ\_SUPPLYINFO, displaying a table with two columns: PNO and QTY. The results are as follows:

PNO	QTY
P1	300
P3	400
P5	400
P1	200
P3	200
P5	100
P3	200

--（2）找出供应商S1的供应情况。

```
SELECT PNO, QTY
```

```
From SJ_SUPPLYINFO
```

```
WHERE SNO = 'S1'
```

The screenshot shows the SQL Server Enterprise Manager interface. The top pane displays the SQL code for creating the view SJ\_SUPPLYINFO. The bottom pane shows the execution results of the query CREATE VIEW SJ\_SUPPLYINFO as select SNO, PNO, QTY FROM SPJ,J WHERE J.JNAME = '三建' and SPJ.JNO = J.JNO;, displaying a table with two columns: SNO and QTY. The results are as follows:

SNO	QTY
P1	300
P3	400
P5	400
P1	200
P3	200
P5	100
P3	200

The screenshot shows the SQL Server Enterprise Manager interface. The top pane displays the SQL code for the query SELECT PNO, QTY FROM SJ\_SUPPLYINFO WHERE SNO = 'S1'. The bottom pane shows the execution results of the query, displaying a table with two columns: PNO and QTY. The results are as follows:

PNO	QTY
P1	300

# 选做：设计学籍管理系统

---

## 一、需求分析

---

学籍数据库的有关语义如下：

1. 一个系可以若干专业，每个专业每年可以招若干班，每个班有若干学生。
2. 系里对每个专业每年都制订了教学计划，学生每年必须按照教学计划修完一定学分的课程（必修课、限选课和任选课），如2000年入学的学生大三上学期必修课30学分，限选课10学分，任选课6学分。
3. 系里的教师可以给多个班带课，但是不能给一个班带多门课程。
4. 一门课程最多允许学生一次补考，学生达到如下条件之一的被开除：
  - 一学期不及格的必修课学分超过10个；
  - 不及格必修课学分累计超过30个；
  - 不及格选修课学分累计超过20个；

1. 查询学生所选修的课程及成绩，并给出必修课平均成绩和选修课平均成绩；

2. 查某一个学生被哪些教师教过课；

3. 查询应被开除的学生（假定差2学分即被开除）。

### 信息要求

需要从数据库中获取 系、教学计划、学生、教师、课程的信息

数据要求：

- 系：系号、名称
- 班：班号、所在系
- 学生：学号、姓名、所在班、课程分数
- 课程：课程号、课程名、课程学分、课程性质
- 教师：姓名，教课班级、教课课程、所属系

### 处理要求

需要完成的处理功能：

- 查询学生所选修的课程及成绩，并给出必修课平均成绩和选修课平均成绩；
- 查某一个学生被哪些教师教过课；
- 查询应被开除的学生（假定差2学分即被开除）

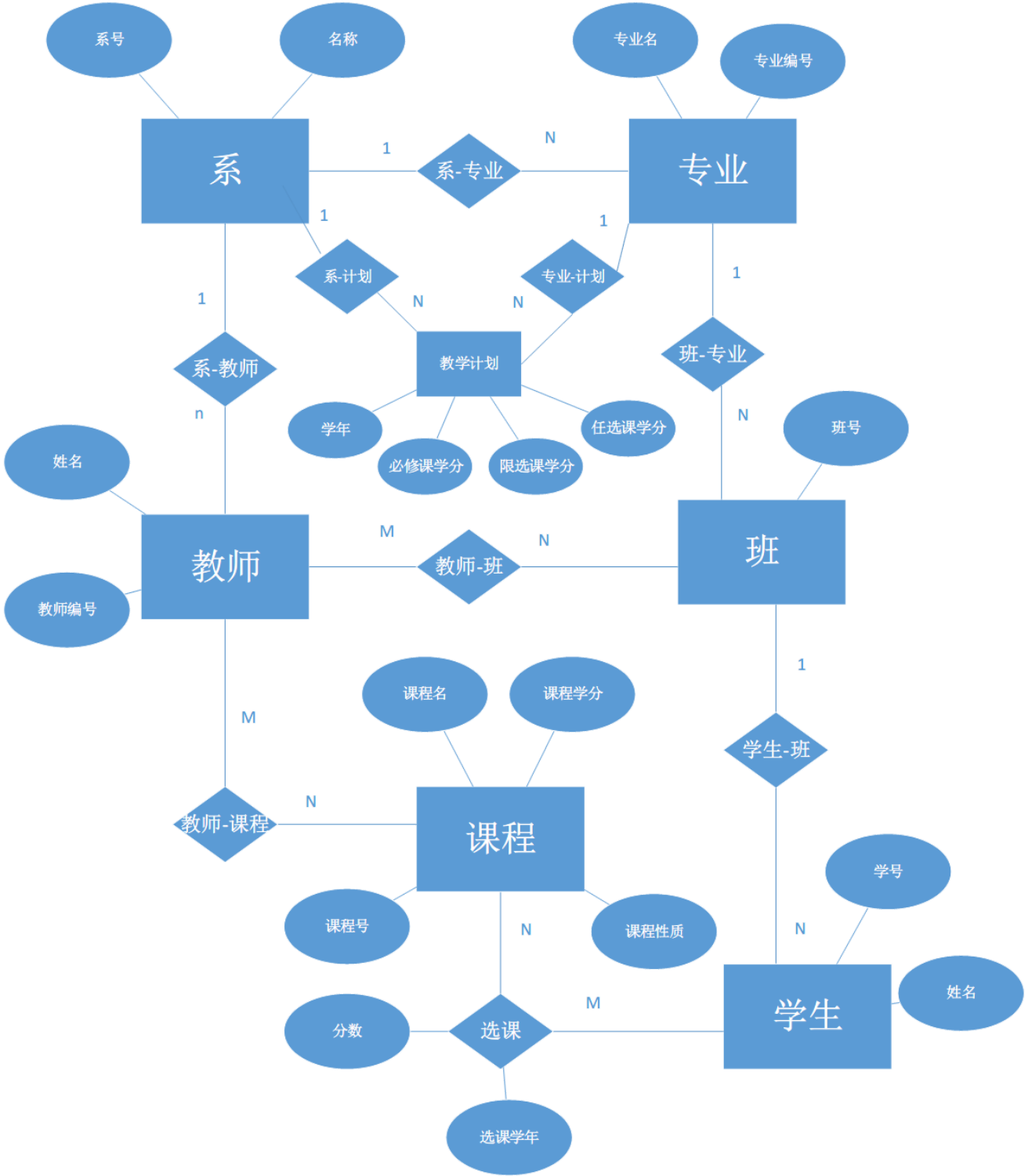
### 安全性和完整性要求

- 相关属性组的非空性和唯一性
- 系里的教师可以给多个班带课，但是不能给一个班带多门课程

## 二、概念结构设计

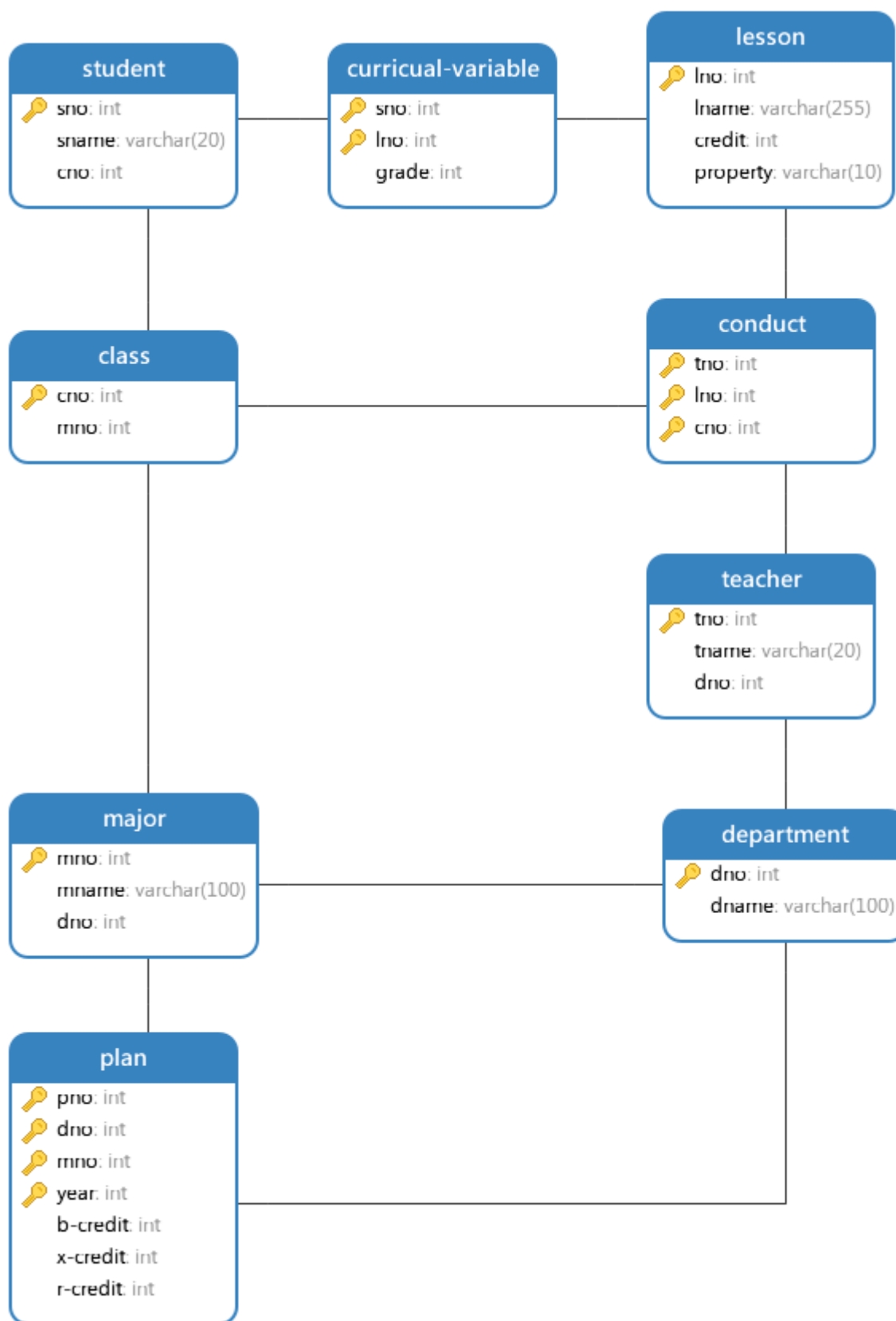
---

根据需求，设计学籍管理系统 E-R 图如下：



### 三、逻辑结构设计

将 E-R 图转换为关系模式如下



## 四、详细实现及源代码

根据建立的关系模式，编写建表 sql 语句



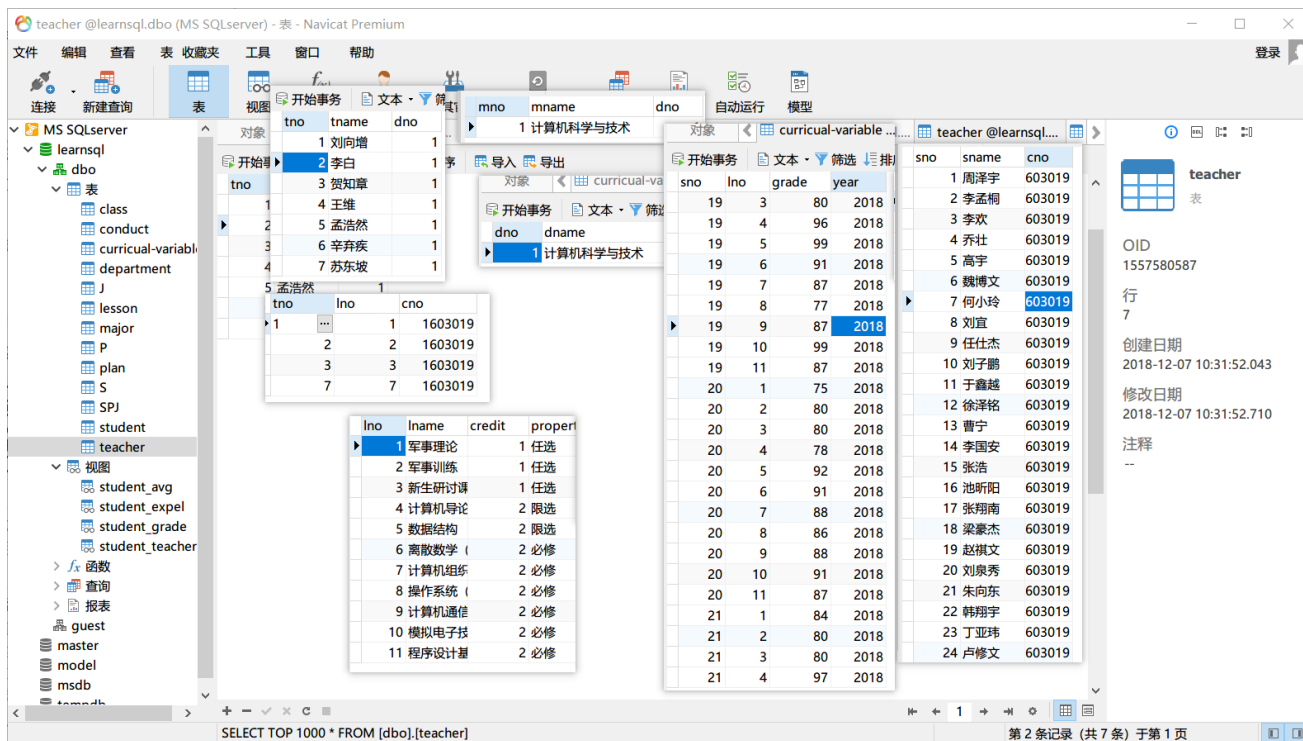
```
1 CREATE TABLE [department] (  
2 [dno] int NOT NULL,  
3 [dname] varchar(100) NOT NULL,  
4 PRIMARY KEY ([dno])  
5 )  
6 GO  
7 CREATE TABLE [major] (  
8 [mno] int NOT NULL,  
9 [mname] varchar(100) NOT NULL,  
10 [dno] int NULL,  
11 PRIMARY KEY ([mno])  
12 )  
13 GO  
14 CREATE TABLE [class] (  
15 [cno] int NOT NULL,  
16 [mno] int NOT NULL,  
17 PRIMARY KEY ([cno])  
18 )  
19 GO  
20 CREATE TABLE [student] (  
21 [sno] int NOT NULL,  
22 [sname] varchar(20) NOT NULL,  
23 [cno] int NOT NULL,  
24 PRIMARY KEY ([sno])  
25 )  
26 GO  
27 CREATE TABLE [teacher] (  
28 [tno] int NOT NULL,  
29 [tname] varchar(20) NOT NULL,  
30 [dno] int NOT NULL,  
31 PRIMARY KEY ([tno])  
32 )  
33 GO  
34 CREATE TABLE [lesson] (  
35 [lno] int NOT NULL,  
36 [lname] varchar(255) NOT NULL,  
37 [credit] int NOT NULL,  
38 [property] varchar(10) NOT NULL,  
39 PRIMARY KEY ([lno])  
40 )  
41 GO  
42 CREATE TABLE [curricula-variable] (  
43 [sno] int NOT NULL,  
44 [lno] int NOT NULL,  
45 [grade] int NOT NULL,  
46 PRIMARY KEY ([sno], [lno])  
47 )  
48 GO  
49 CREATE TABLE [plan] (  
50 [pno] int NOT NULL,  
51 [dno] int NOT NULL,  
52 [mno] int NOT NULL,  
53 [year] int NOT NULL,
```

```

54 [b-credit] int NOT NULL,
55 [x-credit] int NOT NULL,
56 [r-credit] int NOT NULL,
57 PRIMARY KEY ([pno], [dno], [mno], [year])
58 )
59 GO
60 CREATE TABLE [conduct] (
61 [tno] int NOT NULL,
62 [lno] int NOT NULL,
63 [cno] int NOT NULL,
64 PRIMARY KEY ([lno], [cno], [tno])
65 )
66 GO
67
68 ALTER TABLE [major] ADD CONSTRAINT [major-department] FOREIGN KEY ([dno])
REFERENCES [department] ([dno])
69 GO
70 ALTER TABLE [class] ADD CONSTRAINT [class-major] FOREIGN KEY ([mno]) REFERENCES
[major] ([mno])
71 GO
72 ALTER TABLE [student] ADD CONSTRAINT [student-class] FOREIGN KEY ([cno]) REFERENCES
[class] ([cno])
73 GO
74 ALTER TABLE [teacher] ADD CONSTRAINT [teacher-department] FOREIGN KEY ([dno])
REFERENCES [department] ([dno])
75 GO
76 ALTER TABLE [curricual-variable] ADD CONSTRAINT [cur-stu] FOREIGN KEY ([sno])
REFERENCES [student] ([sno])
77 GO
78 ALTER TABLE [curricual-variable] ADD CONSTRAINT [cur-les] FOREIGN KEY ([lno])
REFERENCES [lesson] ([lno])
79 GO
80 ALTER TABLE [plan] ADD CONSTRAINT [plan-dep] FOREIGN KEY ([dno]) REFERENCES
[department] ([dno])
81 GO
82 ALTER TABLE [plan] ADD CONSTRAINT [plan-maj] FOREIGN KEY ([mno]) REFERENCES [major]
([mno])
83 GO
84 ALTER TABLE [conduct] ADD CONSTRAINT [cond-teacher] FOREIGN KEY ([tno]) REFERENCES
[teacher] ([tno])
85 GO
86 ALTER TABLE [conduct] ADD CONSTRAINT [cond-class] FOREIGN KEY ([cno]) REFERENCES
[class] ([cno])
87 GO
88 ALTER TABLE [conduct] ADD CONSTRAINT [cond-lesson] FOREIGN KEY ([lno]) REFERENCES
[lesson] ([lno])
89 GO

```

执行建表语句，并编写测试数据：



可以看到，对应的基本表已经建立，并可以通过基本表建立数据

## 视图的建立

- 建立 `student_grade` 视图，以查询学生的各科成绩

```

1 CREATE VIEW student_grade
2 AS
3 SELECT student.sno, sname,lname,grade,credit,property,[curricual-variable].year
4 FROM student,[curricual-variable],lesson
5 WHERE student.sno = [curricual-variable].sno AND
6       [curricual-variable].[lno] = lesson.lno
7 GO

```

student\_grade @learnsql.dbo (MS SQLServer) - 视图 - Navicat Premium

文件 编辑 查看 表 收藏夹 工具 窗口 帮助

连接 新建查询 表 视图 函数 用户 其它 SQL Server 备份 查询 报表 自动运行 模型

MS SQLServer

- learnsql
  - dbo
    - 表
      - class
      - conduct
      - curricular-variable
      - department
      - J
      - lesson
      - major
      - P
      - plan
      - S
      - SPJ
      - student
      - teacher
    - 视图
      - student\_avg
      - student\_expel
      - student\_grade
      - student\_pass\_cr
      - student\_teacher
    - 函数
    - 查询
    - 报表
    - guest
    - master
    - model
    - model

对象 J @learnsql.dbo (M... P @learnsql.dbo (... plan @learnsql.db... curricular-variable ... student\_grade @le...

开始事务 文本 筛选 排序 导出

sno	sname	lname	grade	credit	property	year
1	周泽宇	军事理论	78	1	任选	2018
1	周泽宇	军事训练	80	1	任选	2018
1	周泽宇	新生研讨课	80	1	任选	2018
1	周泽宇	计算机导论与程序设计	75	2	限选	2018
1	周泽宇	数据结构	96	2	限选	2018
1	周泽宇	离散数学 (I)	79	2	必修	2018
1	周泽宇	计算机组织与体系结构	77	2	必修	2018
1	周泽宇	操作系统 (排课排在通	75	2	必修	2018
1	周泽宇	计算机通信与网络	71	2	必修	2018
1	周泽宇	模拟电子技术基础	81	2	必修	2018
1	周泽宇	程序设计基础课程设计	93	2	必修	2018
2	李孟桐	军事理论	61	1	任选	2018
2	李孟桐	军事训练	80	1	任选	2018
2	李孟桐	新生研讨课	80	1	任选	2018
2	李孟桐	计算机导论与程序设计	98	2	限选	2018
2	李孟桐	数据结构	87	2	限选	2018
2	李孟桐	离散数学 (I)	79	2	必修	2018
2	李孟桐	计算机组织与体系结构	74	2	必修	2018
2	李孟桐	操作系统 (排课排在通	76	2	必修	2018
2	李孟桐	计算机通信与网络	73	2	必修	2018
2	李孟桐	模拟电子技术基础	80	2	必修	2018
2	李孟桐	程序设计基础课程设计	87	2	必修	2018
3	李欢	军事理论	81	1	任选	2018
3	李欢	军事训练	80	1	任选	2018

student\_grade

OID  
2133582639

创建日期  
2018-12-07 20:04:03.047

修改日期  
2018-12-07 20:04:03.047

系统视图  
否

已加密  
否

模式绑定  
否

使用带引号的标识符  
是

使用 Ansi Nulls  
是

有索引  
否

可索引  
否

SELECT TOP 1000 \* FROM [dbo].[student\_grade]

第 1 条记录 (共 330 条) 于第 1 页

- 建立 student\_avg 视图，查询学生的平均成绩

```

1  create view student_avg
2  AS
3  SELECT distinct stu.sno,stu.sname,必修均分,限选均分,任选均分,总均分,必修总学分,限选总学
   分,任选总学分
4  FROM student_grade as stu
5  INNER JOIN (
6      SELECT sno,sname,AVG(grade) as '必修均分',SUM(credit) as '必修总学分'
7      FROM student_grade
8      WHERE property = '必修'
9      group by sno,sname
10 ) as b
11 on stu.sno = b.sno
12 INNER JOIN (
13     SELECT sno,sname,AVG(grade) as '限选均分',SUM(credit) as '限选总学分'
14     FROM student_grade
15     WHERE property = '限选'
16     group by sno,sname
17 )as x
18 on stu.sno = x.sno
19 INNER JOIN (
20     SELECT sno,sname,AVG(grade) as '任选均分',SUM(credit) as '任选总学分'
21     FROM student_grade
22     WHERE property = '任选'
23     group by sno,sname
24 )as r
25 on stu.sno = r.sno
26 INNER JOIN (
27     SELECT sno,sname,AVG(grade) as '总均分',SUM(credit) as '总学分'
28     FROM student_grade
29     group by sno,sname
30 )as z

```

```

31 on stu.sno = z.sno
32 go

```

MS SQLserver | learnsql | dbo | 运行 | 停止 | 解释

```

1 SELECT * FROM [dbo].[student_avg]
2 ORDER BY 总均分 desc

```

信息 结果 1

sno	sname	必修均分	限选均分	任选均分	总均分	必修总学分	限选总学分	任选总学分
25	张俊华	90	100	78	88	12	4	3
19	赵祺文	88	97	80	87	12	4	3
21	朱向东	86	97	81	87	12	4	3
15	张浩	87	98	79	87	12	4	3
4	乔壮	86	96	80	86	12	4	3
24	卢修文	85	96	80	86	12	4	3
20	刘泉秀	88	85	78	85	12	4	3
14	李国安	85	95	78	85	12	4	3
12	徐泽铭	84	90	81	84	12	4	3
18	梁豪杰	85	91	78	84	12	4	3
13	曹宁	79	88	79	81	12	4	3
23	丁亚玮	79	91	80	81	12	4	3
26	李金鑫	79	92	78	81	12	4	3
28	宁寰	80	89	75	80	12	4	3
1	周泽宇	79	85	79	80	12	4	3

SELECT \* FROM [dbo].[student\_avg] ORDER BY 总均分 desc 只读 查询时间: 0.065s

- 建立 student\_pass\_credit 视图，查询学生通过的学分数（成绩>=60分视为通过）

```

1 create view student_pass_credit
2 AS
3 SELECT distinct stu.sno,stu.sname,stu.year,isNULL(bx,0) as 必修通过学
4 分,isNULL(xx,0) as 限选通过学分,isNULL(rx,0) as 任选通过学分
5 FROM student_grade as stu
6 left JOIN (
7     SELECT sno,sname,student_grade.year,SUM(credit) as bx
8     FROM student_grade
9     WHERE property = '必修' and grade >=60
10    group by sno,sname,student_grade.year
11 ) as b
12 on stu.sno = b.sno AND stu.year = b.year
13 left JOIN (
14     SELECT sno,sname,student_grade.year,SUM(credit) as xx
15     FROM student_grade
16     WHERE property = '限选' and grade >=60
17    group by sno,sname,student_grade.year
18 ) as x
19 on stu.sno = x.sno AND stu.year = x.year
20 left JOIN (
21     SELECT sno,sname,student_grade.year,SUM(credit) as rx
22     FROM student_grade
23     WHERE property = '任选' and grade >=60
24    group by sno,sname,student_grade.year

```

```

24 )as r
25 on stu.sno = r.sno AND stu.year = r.year
26 left JOIN (
27     SELECT sno,sname,student_grade.year,SUM(credit) as z
28     FROM student_grade
29     where grade >=60
30     group by sno,sname,student_grade.year
31 )as z
32 on stu.sno = z.sno AND stu.year = z.year
33 go

```

sno	sname	year	必修通过学分	限选通过学分	任选通过学分
1	周泽宇	2018	12	4	3
2	李孟桐	2018	12	4	3
3	李欢	2018	12	4	3
4	乔壮	2018	12	4	3
5	高宇	2018	12	4	3
6	魏博文	2018	12	4	3
7	何小玲	2018	12	4	3
8	刘宜	2018	12	4	3
9	任仕杰	2018	12	4	3
10	刘子鹏	2018	12	4	3
11	于鑫越	2018	12	4	3
12	徐泽铭	2018	12	4	3
13	曹宁	2018	12	4	3
14	李国安	2018	12	4	3
15	张浩	2018	12	4	3
16	池昕阳	2018	4	4	3
17	张翔南	2018	12	4	3
18	梁豪杰	2018	12	4	3
19	赵祺文	2018	12	4	3
20	刘泉秀	2018	12	4	3
21	朱向东	2018	12	4	3
22	韩翔宇	2018	0	2	3
23	丁亚玮	2018	12	4	3
24	卢修文	2018	12	4	3

SELECT TOP 1000 \* FROM [dbo].[student\_pass\_credit]

第 1 条记录 (共 30 条) 于第 1 页

- 建立 student\_expel 视图，筛选需要开除的学生

sno	sname
16	池昕阳
22	韩翔宇

student\_expel 视图

OID  
130099504

创建日期  
2018-12-07 20:29:56.957

修改日期  
2018-12-07 20:29:56.957

系统视图  
否

已加密  
否

模式绑定  
否

使用带引号的识别符  
是

使用 Ansi Nulls  
是

有索引  
否

可索引  
否

- 建立 student\_teacher 视图，查询学生的任课老师和所教课程

```

1 CREATE view student_teacher
2 AS
3 SELECT student.sno,sname,tname,lname
4 FROM student,teacher,conduct,[curricual-variable],lesson
5 WHERE student.cno = conduct.cno AND
6         teacher.tno = conduct.tno and
7         conduct.lno = [curricual-variable].lno and
8         student.sno = [curricual-variable].sno AND
9         lesson.lno = conduct.lno
10 GO

```

对象	student_pass_credi...	SPJ @learnsql.dbo ...	student_avg @lear...	student_pass_credi...	student_teacher @...
开始事务	文本	筛选	排序	导出	
sno	sname	tname	lname		
1	周泽宇	刘向增	军事理论		
1	周泽宇	李白	军事训练		
1	周泽宇	贺知章	新生研讨课		
1	周泽宇	苏东坡	计算机组织与体系结构		
2	李孟桐	刘向增	军事理论		
2	李孟桐	李白	军事训练		
2	李孟桐	贺知章	新生研讨课		
2	李孟桐	苏东坡	计算机组织与体系结构		
3	李欢	刘向增	军事理论		
3	李欢	李白	军事训练		
3	李欢	贺知章	新生研讨课		
3	李欢	苏东坡	计算机组织与体系结构		
4	乔壮	刘向增	军事理论		
4	乔壮	李白	军事训练		
4	乔壮	贺知章	新生研讨课		
4	乔壮	苏东坡	计算机组织与体系结构		
5	高宇	刘向增	军事理论		
5	高宇	李白	军事训练		
5	高宇	贺知章	新生研讨课		
5	高宇	苏东坡	计算机组织与体系结构		
6	魏博文	刘向增	军事理论		
6	魏博文	李白	军事训练		
6	魏博文	贺知章	新生研讨课		
6	魏博文	苏东坡	计算机组织与体系结构		

SELECT TOP 1000 \* FROM [dbo].[student\_teacher]

第 1 条记录 (共 120 条) 于第 1 页

## 建立触发器，实现数据完整性约束

建立 `conduct_instead` 触发器，在插入任课信息前进行完整性检查，即：

- 系里的教师可以给多个班带课，但是不能给一个班带多门课程

```

1 CREATE TRIGGER conduct_instead
2 ON conduct
3 INSTEAD OF update,insert
4 AS
5 IF EXISTS (
6     SELECT * FROM conduct,inserted
7     WHERE conduct.cno = inserted.cno AND
8           conduct.tno = inserted.tno AND
9           NOT EXISTS(
10              SELECT * FROM inserted,deleted
11              WHERE inserted.tno = deleted.tno
12          )
13 )
14 RAISERROR('同一老师不允许给同一个班带多门课程',16,10)
15 ELSE

```

```

16 BEGIN
17     DELETE FROM conduct
18     WHERE exists(
19         select * from deleted
20         WHERE conduct.tno = deleted.tno AND
21             conduct.cno = deleted.cno AND
22             conduct.lno = deleted.lno
23     )
24
25     INSERT into conduct
26     SELECT * FROM inserted
27 end
28 GO

```

定义触发器后，当插入或修改的数据不满足要求时，就会弹出错误警告，并取消插入或修改操作

开始事务 文本 筛选 排序 导入 导出

tno	lno	cno
	1	1 1603019
	2	2 1603019
	3	3 1603019
	7	7 1603019
	3	4 1603019

