建表

```
create table Course
(
    Cno char(4) primary key,
    Cname char(40),
    Cpno char(4),
    Ccredit smallint,
    foreign key (Cpno) references Course(Cno)
)
```

```
create table SC
(
    Sno char(9),
    Cno char(4),
    Grade smallint,
    primary key(Sno,Cno),
    foreign key (Sno) references Student(Sno),
    foreign key (Cno)references Course(Cno)
)
```

course中还没有数据前向SC插入数据导致报错`INSERT 语句与 FOREIGN KEY 约束"FK__SC__Cno__3A81B327"冲突。该冲突发生于数据库"DB_ST"，表"dbo.Course", column 'Cno'。`

course中由于Cpno与Cnp有关联当不存在与Cnp对应的Cpno时会报错`INSERT 语句与 FOREIGN KEY SAME TABLE 约束"FK__Course__Cpno__36B12243"冲突。该冲突发生于数据库"DB_ST"，表"dbo.Course", column 'Cno'。`

因此先将有关联的字段设为空

```
insert into Course (Cno,Cname,Ccredit)
values(2,'数学',2),(3,'信息系统',4),(4,'操作系统',3),(5,'数据结构',4),(6,'数据处理',2),
(7,'PASCSL语言',4);
```

之后再update`update Course set Cpno = 7 where Cno=5;`

`SC表数据插入`

```
insert into SC (Sno,Cno,Grade)
values(201215121,1,92),(201215121,2,85),(201215121,3,88),(201215122,2,90),
(201215122,3,80);
```

select语句

`select Sno,Grade from SC`

`select Sno,Grade from SC where Cno>=2`

`select Sname ,2021-Sage from Student;` 用2021-查询出的Sage显示出的为计算结果

查询字段中有字符串 则字符串会显示出来

```
select Sname ,'test:',2021-Sage from Student;
李勇  test:    2001
刘晨  test:    2002
王敏  test:    2003
张立  test:    2002
```

给列取别名`select Sname studentname,2021-Sage birth from Student;`(给Sname取名叫studentname，给2021-sage取名为birth

`select Sno from SC` 把所用都查出来了（有重复）

`select distinct Sno from SC` 去除了重复的行

and 查询条件 `select * from Student where Sage>18 and Sdept='CS';`

or 查询条件 `select * from Student where Sage>18 or Sdept='CS';`

`or优先级低于and!` `select * from Student where Ssex='男' and Sdept='CS' or Ssex='女' and Sdept='MA';` 实际等于 `where (Ssex='男' and Sdept='CS') or (Ssex='女' and Sdept='MA')`

between: `select * from SC where Grade between 75 and 90;`

in ,not in 查询在不在某个集合里面的数据 `select * from Student where Sdept in ('CS','IS')`
`select * from Student where Sdept not in ('CS','IS')`

like 匹配串（%，_）

`select * from Student where Sdept like '_S';`

`select * from Student where Sno like '2012%'`

当所要匹配的字符串中本身含有（%，_）时需要进行转义 加上（escape ''）这样就可以用\来转义了

`select * from Student where Sno like '2012\%' escape'\';`

is null /is not null 查询为空值的数据，注意不可以用=代替

`select * from Course where Cpno is null ;`

`select * from Course where Cpno is not null ;`

order by 排序

`select * from SC order by Grade ASC ;` //默认升序

`select * from SC order by Grade DESC ;`

聚集函数

count `select count(*) from Course`

`select count(distinct Sno) from SC`

avg (平均值) `select avg(Grade) from SC`

max `select max(Grade) from SC`

sum `select sum(Grade) from SC`

分组查询

```
select Cno,COUNT(Sno) num from SC group by Cno; # 各个课程相应的选课人数用Cno分组

select Sno from SC group by Sno having COUNT(*)>1;
select Sno,avg(Grade) ave from SC  group by Sno having AVG(Grade)>=60;
```

连接查询

```
select Student.*,SC.* from Student,SC where Student.Sno=Sc.Sno;
#将选课信息和学生个人信息链接起来了
```

Sno	Sname	Ssex	Sage	Sdept	Sno	Cno	Grade
201215121	李勇	男	20	CS	201215121	1	92
201215121	李勇	男	20	CS	201215121	2	85
201215121	李勇	男	20	CS	201215121	3	88
201215122	刘晨	女	19	CS	201215122	2	90
201215122	刘晨	女	19	CS	201215122	3	80

去除重复的列就为自然查询

```
select Student.Sno,Sname,Ssex,Sage,Sdept,Cno,Grade from Student,SC where
Student.Sno=Sc.Sno;
```

```
select Student.Sno,Sname from Student,SC where Student.Sno=Sc.Sno and Sc.Cno=2 and
SC.Grade>70;
```

自身连接

- 查询先行课的先行课 first 和second是人为起的名字

```
Select first.Cno,second.CPno from Course first ,Course second where
first.Cpno=second.Cno;
```

外连接

- 左外链接（左右看from表的顺序）

```
Select Student.Sno,Sname,Ssex,Sage,Sdept,Cno,Grade from Student left outer join SC on
(Student.Sno=SC.Sno)
```

- 右外连接

```
Select Student.Sno,Sname,Ssex,Sage,Sdept,Cno,Grade from Sc right outer join Student
on (Student.Sno=SC.Sno)
```

多表连接

```
select Student.Sno,Sname,Cname,Grade from Student ,Course,SC where Student.Sno=Sc.Sno and
SC.Cno=Course.Cno
```

嵌套查询

```
select Sname from Student where Sno in (select Sno from SC where Cno=2)
```

```
select Sno,Sname,sdept from Student where Sdept = (select Sdept from Student where
Sname='刘晨')
```

```
select Sno,Cno,Grade from SC x where Grade >=(select AVG(Grade) from SC y where
x.Sno=y.Sno)
```

`x y是给表起的别名`

带有谓词的子查询

`all 所有 any 一些`

```
select Sname,Sage from Student where Sage<All (select Sage from Student where Sdept='CS')

select Sname,Sage from Student where Sage<Any (select Sage from Student where Sdept='CS')
```

- exists 谓词查询 不返回数据只返回真假

```
select Sname from Student where exists (select * from SC where SC.Sno=Student.Sno and
Cno=1)
```

- not exists(未选修改门课程的学生)

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

- 所有的其他谓词（in any all等）以及比较运算符 都可以用exists来替换，但是不是所有的exists都可以被其他写法替代
- 有点难以理解的查询 查找选修了全部课程的学生 方法：没有一门课程是这个学生没有选的课 没有没有选的课

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

集合查询

- union 去除重复项 如果想要保留则需要用 union all

```
select * from student where Sdept='CS'
union
```

```
select * from student where Sage<=19


select * from student where Sdept='CS'
union all
select * from student where Sage<=19
```

- intersect 交集

```
select * from student where Sdept='CS'
intersect
select * from student where Sage<=19
```

- except 差集

```
select * from student where Sdept='CS'
except
select * from student where Sage<=19
```

派生表

```
select Sno,Cno from
SC,(select Sno,avg(Grade) from SC group by Sno)
as avg_sc(avg_sno,avh_grade)
where
    Sc.sno=avg_sc.avg_sno and SC.Grade>=avg_sc.avh_grade
```

数据更新

```
insert into Student(Sno,Sname,Ssex,Sdept,Sage)
values ('201215128','陈冬','男','IS',18);
```

- 插入子查询结果

  首先建立一个表

```
create table Dept_age
(Sdept char(15),
Avg_age smallint)
```

然后插入选择后的数据

```sql
insert
into
Dept_age(Sdept,Avg_age)
select Sdept,avg(Sage)
from Student
group by Sdept
```

更新数据

```sql
update Student
set Sage=22
where Sno='201215121'
```

删除数据

```sql
delete Student
where Sno='201215121'
```

空值的处理

构建空值

```sql
insert into SC(Sno,Cno,Grade)
    values('201215125','1',NULL)
```


空值的判断

```sql
select * from Student where Sname is null or Ssex is null or Sage is null or Sdept is
null
```

视图

- 创建视图

```sql
create view IS_Student
as
select * from Student where Sdept='CS'
```

- 如果添加 `with chect option` 则之后对视图进行修改 插入和删除时 系统会继续保持视图建立时的选择条件

```sql
create view IS_Student
as
select * from Student
where Sdept='IS'with check option
```

```sql
create view IS_S1(Sno,Sname,Grade)
as select Student.Sno,Sname,Grade
from Student,SC
where Sdept='IS' and Student.Sno=SC.Sno and SC.Cno=1
```

带有虚拟列的视图

```sql
create view BT_S(Sno,Sname,Sbirth)
as
select Sno,Sname,2014-Sage
from Student
```

删除视图

```sql
drop view BT_S;
drop view BT_S cascade;
```

后面的关键字会将由此视图导出的所有视图都删掉（级联删除）

查询视图

```sql
select IS_Student.Sno,Sname
from IS_Student,SC
where IS_Student.Sno=Sc.Sno and Sc.Cno=1
```

```sql
select *
from(
select Sno,avg(Grade) from SC  group by Sno) as S_G(Sno,Gavg)
where Gavg>=80
```

更新视图

```sql
update IS_Student
set Sname='刘辰'
where Sno='201215122' and Sdept='IS'
```

插入

```sql
insert into IS_Student (Sno,Sname,Sage,Sdept)
values('201215129','赵新','20','IS')
```

```

删除

```sql
delete from IS_Student where Sno='201215129'
```

注意 以下操作无法完成 原因是平均成绩是根据多科成绩算来的 系统无法实现更改多科成绩使得平均成绩满足要求

```sql
create view S_G(Sno,Gavg)
AS
select Sno,avg(Grade) from SC group by Sno;

update S_G
set Gavg=90
where Sno='201215121'
```

数据库安全性

注意此部分在sqlserver中不需要加table

GRANT授权

```sql
GRANT <权限> [<权限>]···
ON <对象类型><对象名> [,<对象类型><对象名>]···
TO<用户>[,<用户>]···
[WITH GRANT OPTION];
```

`with grant option` 有权限将此权限授予他人 可传播

示例：

```sqlserver
grant select
on Student
to U1;
```

```sqlserver
grant select
on SC
to public;
```

```sql
grant update(Sno),select
on Student
to U4;
```

REVOKE 收回权限

```sql
REVOKE <权限>[,<权限>]···
ON <对象类型><对象名>[,<对象类型><对象名>]···
FROM <用户>[,<用户>]···[CASCADE|RESTRICT];
```
```

```

示例：

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

```sqlserver
revoke insert
on SC
from U5 cascade;
```

数据库角色

用creat role创建 可用grant和revoke来授权和收回。

```sql
create role R1;
```



```sql
grant select,update,insert
on Student
to R1;
```



```sql
grant delete
on Student
to R1;
```



```sql
revoke select
on Student
from R1;
```

完整性约束

实体完整性

两种定义方式

- 列级定义
- 表级定义

```sql
列级
create table Student
(Sno char(9) primary key, -
```
```

```
 Sname char(20) unique,
 Ssex char(2),
 Sage smallint,
 Sdept char(20)
);
```

```sql
表级
create table Student
(Sno char(9),
 Sname char(20) not null,
 Ssex char(2),
 Sage smallint,
 Sdept char(20),
 primary key(Sno)
);
```

当主码有两个以上属性时只能表级定义

```sql
create table SC
(Sno char(9)not null,
 Cno char(4)not null,
 Grade smallint,
 primary key(Sno,Cno),
);
```

参照完整性（外码）

```sql
create table SC
(Sno char(9)not null,
 Cno char(4)not null,
 Grade smallint,
 primary key(Sno,Cno),
 foreign key(Sno)references Student(Sno),-
 foreign key(Cno)references Course(Cno) -
);
```


用户自定义的完整性

非空

```sql
Cno char(4)not null,
```

唯一（不可重复）

```sql
Dname char(9) unique not null
```

check自定义条件

```sql
create table Student
(Sno char(9) primary key,
 Sname char(20) not null,
 Ssex char(2) check (Ssex in('男','女')),
 Sage smallint,
 Sdept char(20)
);
```

```sqlserver
create table SC
(Sno char(9),
 Cno char(4),
 Grade smallint check(Grade>=0 and Grade<=100),
 primary key(Sno,Cno),
 foreign key(Sno)references Student(Sno),
 foreign key(Cno)references Course(Cno)
);
```

练习题

`某单位想矩形一个小型的联谊会，关系Male'记录注册的男宾信息，Femal记录注册的女宾信息。建立一个断言讲来宾的人数限制在50以内`

首先建立两张表

```sql
create table Female
(
    Name varchar(10),
    Age int
);
create table Male
(
    Name varchar(10),
    Age int
)
```

建立断言

```sql
create assertion demo check (60>=select count(*) from Male +select count(*) from Femal);
```

但是发现一个问题--sqlserver本身并不支持断言。。。。。