连接查询

等值与非等值连接查询

查询选修2号课程且成绩在90分以上的所有学生的学好和姓名

```
SELECT Student.Sno,Sname
FROM Studnet,SC
WHERE Student.Sno=SC.Sno AND SC.Cno='2' AND SC.Grade > 90;
```

OR

```
SELECT Student.Sno,Sname
FROM Studnet JOIN SC ON Student.Sno=SC.Sno
WHERE SC.Cno='2' AND SC.Grade > 90;
```

$$\pi_{Sno,Sname}(\sigma_{Cno='2' grade > 90}(SC) \infty Studnet)$$

自身连接

一个表与自己连接

查询每一门课的间接先行课

```
SELECT FIRST.Cno,SECOND.Cno
FROM Course FIRST,Course SECOND
WHERE
```

外连接

普通连接操作只输出满足条件的

查询每个学生及其选修课的情况，用左外连接

```
SELECT Student.Sno,Sname,Ssex,Sage,Sdept,Cno,Grade
FROM Student LEFT OUTER JOIN SC ON (Student.Sno = SC.Sno);
```

1.查询所有选课学生的成绩单，包括学生的学号，姓名，学修课程名和成绩

```
SELECT Student.Sno,Sname,Course.Cno,Grade
FROM Student,SC,Course
WHERE Student.Sno=SC.Sno,SC.Cno=Course.Cno;
```

1.查 ***所有学生*** 的成绩单，包括学生的学号，姓名，学修课程名和成绩

```
SELECT Student.Sno,Sname,Course.Cno,Grade
FROM Student LEFT OUTER JOIN SC ON (Student.Sno=SC.Sno) LEFT OUTER JOIN Course ON
(SC.Cno=Course.Cno);
```

2.生成所有学生的平均成绩单，包括姓名和平均成绩

```
SELECT Student.Sname,AVG(SC.Grade)
FROM SC
GROUP BY SNO
LEFT OUTER JOIN Student ON (Student.Sno=SC.Sno)
```

3.选课人数超过１００的课程编号及选课人数

```
SELECT Course.Cno,COUNT(*)
FROM SC
GROUP BY Cno
WHERE HAVING COUNT(*) > 100;
```

4."数据库"课程的先修课的课程编号，课程名和学分

```
SELECT
```

嵌套查询

```
SELECT Sname
FROM Student
WHERE Sno IN{
  SELECT Sno
  FROM SC
  WHERE Cno='c2'
}
```

- 不相关子查询：查询条件不依赖与主查询
- 相关子查询：处理效率较低

带有IN谓词的子查询

查询与刘晨在同一个系学习的学生

```
SELECT Sname
FROM Student
WHERE Sdept IN
    (SELECT Sdept
     FROM Studnet
     WHERE Sname='刘晨')；
```

查询选修信息系统的学生学号和姓名

```
SELECT Sno, Sname
FROM Studnet
WHERE Sno IN
     (SELECT Sno
      FROM SC
      WHERE Cno IN
           (SELECT Cno
            FROM Course
            WHERE Cname='OS'))
```

查询订购了产品号‘RG1’的顾客编号

```
SELECT cust_id
FROM Order
WHERE order_id IN
      (SELECT order_id
       FROM OrderItem
       WHERE prod_id='RG1');
```

带有比较运算符的子查询

如果明确知道是单行单列的关系可以作为单值使用

```
SELECT Sname
FROM Student
WHERE Sdept ＝   //可以用＝代替IN
   (SELECT Sdept
    FROM Studnet
    WHERE Sname='刘晨')；
```

查询每个学生超过他选修课平均分的课程号

```
SELECT Sno,Cno
FROM SC x
WHERE Grade >=(SELECT AVG(Grade)
                FROM SC y
              WHERE y.Sno=x.Sno)
```

找出定价高于所有产品平均价格的产品的名称

```
SELECT name
FROM Product
WHERE price >=(SELECT AVG(price)
                FROM Product)
```

带有ANY（SOME）或ALL谓词的子查询

使用ANY或ALL谓词时必须使用比较运算

```
> < >= <= = ALL
> < >= <= = ANY
```

查询非计算机系中年龄小于计算机系中任意一个学生年龄的学生姓名和年龄

```
SELECT Sname,Sage
FROM Student
WHERE SAGE < ANY (SELECT Sage
                  FROM Student
                  WHERE Sdge = 'CS')
WHERE Sdge <> 'CS'
```

用聚集函数实现

```
SELECT Sname,Sage
FROM Student
WHERE SAGE < ANY (SELECT MAX(Sage)
                  FROM Student
                  WHERE Sdge = 'CS')
WHERE Sdge <> 'CS'
```

查询非计算机系中年龄小于计算机系中所有学生年龄的学生姓名和年龄

```
SELECT Sname,Sage
FROM Student
WHERE SAGE < ALL (SELECT Sage
                  FROM Student
                  WHERE Sdge = 'CS')
WHERE Sdge <> 'CS'
```

供应商‘V1’ 提供了大量商品，请找出其他供应商提供的定价低于‘V1’所有产品价格的产品编号，供应商编号和名称。

```
SELECT prod_id,vend_id,name
FROM Product
WHERE price < ALL (SELECT price
                   FROM Product
                   WHERE vend_id='V1')
WHERE vend_id <> 'V1'
```

带有否定含义的子查询

查询没有选修 1 号课程的学生姓名

```
SELECT Sname
FROM Student
WHERE Sno NOT IN (SELECT Sno
                  FROM Student
                  WHERE SC='1')
```

带有 exists字句的子查询

查询与刘晨在同一个系的学生

```
SELECT Sname
FROM Student S1
WHERE EXISTS (SELECT *
              FROM Student S2
              WHERE S1.Sdge=S2.Sdge
              AND Sname='刘晨')
```

使用exists语句查询购买了编号为001产品的顾客编号

```
SELECT cust_id
FROM Order
WHERE EXISTS (SELECT prod_id
              FROM OrderItem
              WHERE OrderItem.order_id=Order.order_id
              AND prod_id='001')
```

查询只选修了课程‘C01’的学生学号

```
SELECT Sno
FROM SC A
WHERE A.Cno='C01'
AND NOT EXISTS(WHERE )
```

将操作系统课程的成绩全部设为0

```
UPDATE SC SET GRADE=0
WHERE Cno IN (WHERE Cno
              FROM Course
              WHERE Cname ='OSS')
```

集合查询

集合操作的种类

- 并UNION

```
UNION ALL：保留重复元祖
UNION：不保留
```

- 交INTERSECT
- 差MINUS