第1章 多表查询

1.多表查询类型

1.1 笛卡尔乘积

```
select * from teacher,course;
或者:
select * from teacher join course;

拿着 teacher每行数据和course逐行进行组合，显示
两层for循环的实现逻辑。Simple-Nextloop （嵌套循环方式）
得出的结果，会有部分数据是无意义的。
```

1.2 内连接 join 取交集

```
mysql> select * from teacher join course on teacher.tno=course.tno ;
+-----+--------+------+--------+-----+
| tno | tname  | cno  | cname  | tno |
+-----+--------+------+--------+-----+
| 101 | oldboy | 1001 | linux  | 101 |
| 102 | hesw   | 1002 | python | 102 |
| 103 | oldguo | 1003 | mysql  | 103 |
+-----+--------+------+--------+-----+
3 rows in set (0.00 sec)

mysql> select * from teacher,course where teacher.tno=course.tno;
+-----+--------+------+--------+-----+
| tno | tname  | cno  | cname  | tno |
+-----+--------+------+--------+-----+
| 101 | oldboy | 1001 | linux  | 101 |
```

```
| 102 | hesw   | 1002 | python | 102 |
| 103 | oldguo | 1003 | mysql  | 103 |
+-----+--------+------+--------+-----+
3 rows in set (0.00 sec)
```

1.3 外连接 left join , right join

```
mysql> select * from teacher left join course on
teacher.tno=course.tno;
+-----+--------+------+--------+------+
| tno | tname  | cno  | cname  | tno  |
+-----+--------+------+--------+------+
| 101 | oldboy | 1001 | linux  |  101 |
| 102 | hesw   | 1002 | python |  102 |
| 103 | oldguo | 1003 | mysql  |  103 |
| 104 | oldx   | NULL | NULL   | NULL |
| 105 | oldw   | NULL | NULL   | NULL |
+-----+--------+------+--------+------+
5 rows in set (0.00 sec)

mysql> select * from teacher right join course on
teacher.tno=course.tno;
+------+--------+------+--------+-----+
| tno  | tname  | cno  | cname  | tno |
+------+--------+------+--------+-----+
|  101 | oldboy | 1001 | linux  | 101 |
|  102 | hesw   | 1002 | python | 102 |
|  103 | oldguo | 1003 | mysql  | 103 |
| NULL | NULL   | 1004 | k8s    | 108 |
+------+--------+------+--------+-----+
4 rows in set (0.00 sec)
```

2.多表连接语法

2.1 a表 和 b表 有直接的关联关系

```
select a.x,b.y  from a  join b on a.z=b.z  where  group by  having  order by limit;

select a.x,b.y    #查找的内容
from a
join b            #a关联b
on a.z=b.z        #关联条件
where             #其他条件
group by          #分组依据
having            #分组后判断
order by          #排序规则
limit;            #显示条目
```

2.2 a表 和 b表 没有直接的关联关系

```
假如: a和c 有关, b和c有关

a join c on a.i = c.j join b on c.x=b.y

a join c
on a.i = c.j
join b
on c.x=b.y
```

2.3 套路

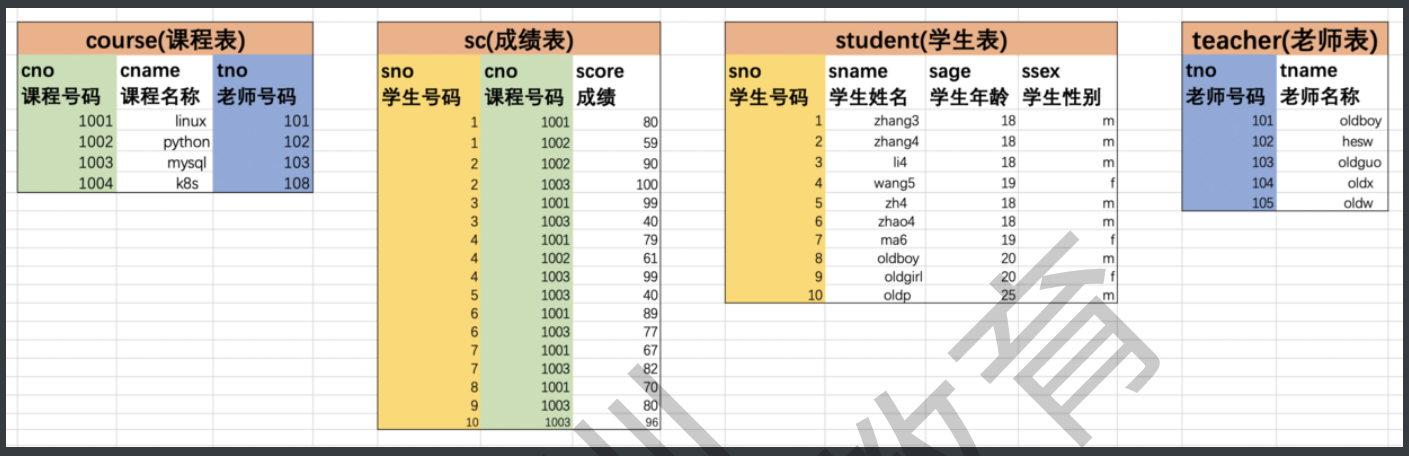
```
1.根据题意将所有涉及到的表找出来 a b
2.找到a和b直接或者间接的关联条件
3.用join on 语句把所有表连接到一起
4.罗列其他查询条件
```

3.大量练习

3.1 导入数据

```
source /root/school.sql
```

关系图:



course(课程表)

cno 课程号码	cname 课程名称	tno 老师号码
1001	linux	101
1002	python	102
1003	mysql	103
1004	k8s	108

sc(成绩表)

sno 学生号码	cno 课程号码	score 成绩
1	1001	80
1	1002	59
2	1002	90
2	1003	100
3	1001	99
3	1003	40
4	1001	79
4	1002	61
4	1003	99
5	1003	40
6	1001	89
6	1003	77
7	1001	67
7	1003	82
8	1001	70
9	1003	80
10	1003	96

student(学生表)

sno 学生号码	sname 学生姓名	sage 学生年龄	ssex 学生性别
1	zhang3	18	m
2	zhang4	18	m
3	li4	18	m
4	wang5	19	f
5	zh4	18	m
6	zhao4	18	m
7	ma6	19	f
8	oldboy	20	m
9	oldgirl	20	f
10	oldp	25	m

teacher(老师表)

tno 老师号码	tname 老师名称
101	oldboy
102	hesw
103	oldguo
104	oldx
105	oldw

3.2 每位老师所教课程名称

```
select
teacher.tname,course.cname
from teacher
join course
on teacher.tno=course.tno;
```

3.3 统计每个学员，学习课程的门数

```
select student.sname'学生姓名',COUNT(*)'学习门数'
from student
join sc
on student.sno=sc.sno
GROUP BY student.sno;
```

3.4 统计每个学员，学习课程的门数和课程名列表

关系图:

```
student ----> sc  ----> course ---> teacher
```

语句:

```
select
CONCAT(student.sname,"_",student.sno),COUNT(*),GROUP_CONCAT(course.cname
)
from student
join sc
on student.sno=sc.sno
join course
on sc.cno=course.cno
group by student.sno
```

3.5 每位老师教的学生数量和学生名列表

关系图:

```
student ----> sc  ----> course ---> teacher
```

语句:

```
select
CONCAT(teacher.tname,"_",teacher.tno),COUNT(*),GROUP_CONCAT(student.snam
e)
from teacher
join course
on teacher.tno=course.tno
join sc
on course.cno=sc.cno
join student
on sc.sno=student.sno
group by teacher.tno
```

3.6 每位老师教所教课程的平均分

```
select
CONCAT(teacher.tname,"_",teacher.tno,"_",course.cno),AVG(sc.score)
from teacher
join course
on teacher.tno=course.tno
join sc
on course.cno=sc.cno
group by teacher.tno , course.cno
```

3.7 查找学习了hesw但没学习oldguo课程的学生名

case用法：

```
select case when 1=1 then "true" end

USE mysql;
SELECT
case
WHEN USER='root' THEN HOST END,
WHEN USER !='root' THEN 2 END
FROM mysql.user;
```

```
9
10 USE mysql;
11 SELECT
12 CASE
13 WHEN USER='root' THEN HOST
14 WHEN USER !='root' THEN 2 END
15 FROM mysql.user;
```

方法1:

```
1  select a.sname from
2  a
3  left join
4  b
5  on a.sname=b.sname
6  where b.sname is null;
7
8  select a.sname from
9  (select student.sname
10 from teacher
11 join course
12 on teacher.tno=course.tno
13 join sc
14 on course.cno=sc.cno
15 join student
16 on sc.sno=student.sno
17 where teacher.tname = 'hesw') as a
18 left join
19 (select student.sname
20 from teacher
21 join course
22 on teacher.tno=course.tno
23 join sc
24 on course.cno=sc.cno
25 join student
```

```
on sc.sno=student.sno
where teacher.tname = 'oldguo') as b
on a.sname=b.sname
where b.sname is null
```

方法2:

```
SELECT student.`sname`,GROUP_CONCAT(teacher.`tname`)
FROM course
JOIN sc
ON course.cno=sc.cno
JOIN student
ON sc.sno=student.sno
JOIN teacher
ON course.tno=teacher.tno
GROUP BY student.sname
HAVING GROUP_CONCAT(teacher.`tname`) LIKE '%hesw%' AND
GROUP_CONCAT(teacher.`tname`) NOT LIKE '%oldguo%';
```

3.8 查询出只选修了一门课程的全部学生的学号和姓名

```
SELECT student.sname,student.sno,COUNT(sc.cno)
FROM sc
JOIN student
ON sc.sno=student.sno
GROUP BY sc.sno
HAVING COUNT(sc.cno)=1;
```

3.9 查询各科成绩最高和最低的分：以如下形式显示：课程名称，最高分，最低分

```
SELECT course.cname'课程名称',MAX(sc.`score`)'最高分',MIN(sc.`score`)'最低分'
FROM sc
JOIN course
ON sc.cno=course.cno
GROUP BY course.cname;
```

3.10 查询平均成绩大于85的所有学生的学号、姓名和平均成绩

```
select sc.sno,student.sname,AVG(sc.score)
from sc
join student
on sc.sno=student.sno
group by sc.sno
having AVG(sc.score)>85;
```

3.11 统计每门课程:优秀(85分以上),良好(70-85),一般(60-70),不及格(小于60)的学生列表

```
select
course.cname ,
GROUP_CONCAT(case when sc.score>=85 then  student.sname end),
GROUP_CONCAT(case when sc.score>=70 and sc.score<85 then  student.sname end),
GROUP_CONCAT(case when sc.score>=60 and sc.score<70 then  student.sname end),
GROUP_CONCAT(case when sc.score<60 then  student.sname end)
from course
join sc
on course.cno=sc.cno
join student
on sc.sno=student.sno
group by course.cno

SELECT course.cname,
```

```
15  GROUP_CONCAT(CASE WHEN sc.score>=85 THEN
    CONCAT(student.sname,":",sc.score) END)'优秀',
16  GROUP_CONCAT(CASE WHEN sc.score>=75 AND sc.`score`<85 THEN
    CONCAT(student.sname,":",sc.score) END)'良好',
17  GROUP_CONCAT(CASE WHEN sc.score>=60 AND sc.`score`<75 THEN
    CONCAT(student.sname,":",sc.score) END)'一般'
18  FROM course
19  JOIN sc
20  ON course.cno=sc.cno
21  JOIN student
22  ON sc.sno=student.sno
23  GROUP BY course.cname;
```

3.12 表别名使用

```
1   select
2   a.cname ,
3   GROUP_CONCAT(case when b.score>=85 then  c.sname end),
4   GROUP_CONCAT(case when b.score>=70 and b.score<85 then  c.sname end),
5   GROUP_CONCAT(case when b.score>=60 and b.score<70 then  c.sname end),
6   GROUP_CONCAT(case when b.score<60 then  c.sname end)
7   from course as a
8   join sc as b
9   on a.cno=b.cno
10  join student as c
11  on b.sno=c.sno
12  group by a.cno
```

3.13 列别名

```
select
a.cname as "课程名称" ,
GROUP_CONCAT(case when b.score>=85 then  c.sname end) as "优秀学员",
GROUP_CONCAT(case when b.score>=70 and b.score<85 then  c.sname end) as "良好学员",
GROUP_CONCAT(case when b.score>=60 and b.score<70 then  c.sname end) as "一般学员",
GROUP_CONCAT(case when b.score<60 then  c.sname end) as "不及格学员"
from course as a
join sc as b
on a.cno=b.cno
join student as c
on b.sno=c.sno
group by a.cno
```

第2章 元数据获取

1.常用show语句

```
help show;
show databases;                 # 查询所有库名
show tables;                    # 查询当前库的所有表名
show tables from world;         # 查询world库下的所有表名
show create database world;       # 查询world建库语句
show create table city;           # 当前库下的city表建表语句
show create table world.city;     # world库下的建表语句
show privileges;                  # 数据库中所有权限
show engines;                     # 数据库中支持的存储引擎
show grants for root@'localhost'  # 查询某个用户权限
show charset;                       # 查询数据库字符集支持
show collation;                     # 查询所有校对规则的支持
show variables like '%trx%'         # 查询数据库参数
```

```
show status like 'com_%'              # 查询数据库的状态
show processlist;                    # 查询所有会话信息
show engine innodb status             # 查询innodb引擎相关的状态
show binary logs                      # 查询二进制日志文件信息
show binlog events in 'xxx'           # 查看二进制日志事件
show master status ;                  # 当前正在使用的二进制日志信息
show slave status\G                   # 查看主从状态相关信息
show slave hosts;                     # 查看从库主机信息
```