MySQL 基本用法(二)多表查询

数据库

创建表并初始化数据

1、表说明

student(学生表)、teacher(教师表)、course(课程表)、sc(分数表)

2、创建表

```
/*班级表*/
CREATE TABLE `class` (
`cid` int(11) NOT NULL AUTO_INCREMENT,
`cname` varchar(32) NOT NULL,
PRIMARY KEY (`cid`)
)

/*学生表*/
create table student(
   sno varchar(10) primary key,
   sname varchar(20),
   sage float(2),
   ssex varchar(5),
   `class_id` int(11) NULL
);
/*教师表*/
create table teacher(
   tno varchar(10) primary key,
   tname varchar(20)
);
/*课程表*/
create table course(
   cno varchar(10),
   cname varchar(20),
   tno varchar(20),
   constraint pk_course primary key (cno,tno)
);
/*分数表*/
create table sc(
   sno varchar(10),
   cno varchar(10),
   score float(4,2),
   constraint pk_sc primary key (sno,cno)
);

```

3、插入初始化数据

```
/***************班级表***********************/
insert into `class` values (1,'1班');
insert into `class` values (2,'2班');
insert into `class` values (3,'3班');

/*******初始化学生表的数据******/
insert into student values ('s001','张三',23,'男',1);
insert into student values ('s002','李四',23,'男',1);
insert into student values ('s003','吴鹏',25,'男',2);
insert into student values ('s004','琴沁',20,'女',1);
insert into student values ('s005','王丽',20,'女',2);
insert into student values ('s006','李波',21,'男',1);
insert into student values ('s007','刘玉',21,'男',3);
insert into student values ('s008','萧蓉',21,'女',1);
insert into student values ('s009','陈萧晓',23,'女',2);
insert into student values ('s010','陈美',22,'女',3);

/*****************初始化教师表***********************/
insert into teacher values ('t001', '刘阳');
insert into teacher values ('t002', '谌燕');
insert into teacher values ('t003', '胡明星');

/***************初始化课程表*****************************/
insert into course values ('c001','J2SE','t002');
insert into course values ('c002','Java Web','t002');
insert into course values ('c003','SSH','t001');
insert into course values ('c004','Oracle','t001');
insert into course values ('c005','SQL SERVER 2005','t003');
insert into course values ('c006','C#','t003');
insert into course values ('c007','JavaScript','t002');
insert into course values ('c008','DIV+CSS','t001');
insert into course values ('c009','PHP','t003');
insert into course values ('c010','EJB3.0','t002');

/***************初始化成绩表***********************/
insert into sc values ('s001','c001',78.9);
insert into sc values ('s002','c001',80.9);
insert into sc values ('s003','c001',81.9);
insert into sc values ('s004','c001',60.9);
insert into sc values ('s001','c002',82.9);
insert into sc values ('s002','c002',72.9);
insert into sc values ('s003','c002',81.9);
insert into sc values ('s001','c003','59');
```

数据完整性

作用：保证用户输入的数据保存到数据库中是正确的。
确保数据的完整性 = 在创建表时给表中添加约束
完整性的分类：

```
实体完整性
域完整性
引用完整性
```

1.1 实体完整性

实体: 即表中的一行(一条记录)代表一个实体(entity)
实体完整性的作用：标识每一行数据不重复
约束类型： 主键约束（primary key） 唯一约束(unique) 自动增长列(auto_increment)
1.1.1 主键约束

注：每个表中要有一个主键。
特点：数据唯一，且不能为null

例子:
第一种添加方式:

```
CREATE TABLE student(
id int primary key,
name varchar(50)
);
```

第二种添加方式:

```
CREATE TABLE student(
id int,
name varchar(50),
primary key(id)
);
CREATE TABLE student(
classid int,
stuid int,
name varchar(50),
primary key(classid，stuid)
);
```

此种方式优势在于，可以创建联合主键

第三种方式:

```
CREATE TABLE student(
id int,
name varchar(50)
);
ALTER TABLE student  ADD  PRIMARY  KEY (id);
```

1.1.2 唯一约束(unique):

特点: 数据不能重复

```
CREATE TABLE student(
Id int primary key,
Name varchar(50) unique
);
```

1.1.3

自动增长列(auto_increment)

> 这种用法只限于mysql,其他数据库略有不同

```
CREATE TABLE student(
Id int primary key auto_increment,
Name varchar(50)
);
INSERT INTO student(name) values('tom');
```

1.2 域完整性

域完整性的作用：限制此单元格的数据正确，不对照此列的其它单元格比较

域代表当前单元格
域完整性约束：数据类型 非空约束（not null） 默认值约束(default)
check约束（mysql不支持）check(sex='男' or sex='女')
1.2.1 非空约束: not null

```
CREATE TABLE student(
Id int pirmary key,
Name varchar(50) not null,
Sex varchar(10)
);

INSERT INTO student values(1,'tom',null);
```

1.2.2 默认值约束 default

```
CREATE TABLE student(
Id int pirmary key,
Name varchar(50) not null,
Sex varchar(10) default '男'
);

insert into student1 values(1,'tom','女');
insert into student1 values(2,'jerry',default);
```

1.3 引用完整性

外键约束(实际开发中基本上不会使用)

例:

```
CREATE TABLE student(
sid int pirmary key,
name varchar(50) not null,
sex varchar(10) default '男'
);

create table score(
    id int,
    score int,
    sid int , -- 外键列的数据类型一定要与主键的类型一致
    CONSTRAINT fk_score_sid foreign key (sid) references student(id)
);
```

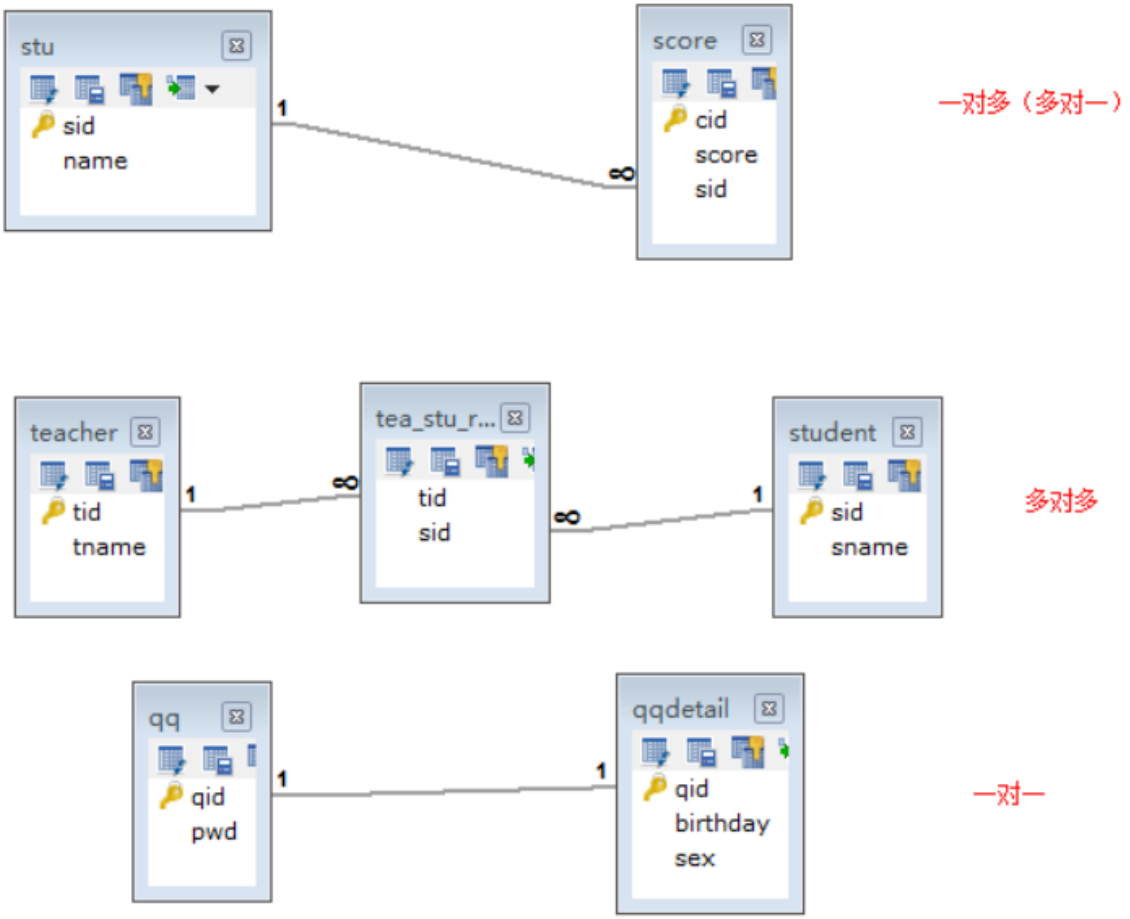
第二种添加方式

```
ALTER TABLE score ADD CONSTRAINT fk_stu_score FOREIGN KEY(sid) REFERENCES
stu(id);
```

外键会保证score中的数据是student表中sid这一列已经存在的值

表与表之间的关系

```
一对一：例如t_person表和t_card表，即人和身份证。这种情况需要找出主从关系，即谁是主表，谁是从表。人可以没有身份证，但身份证必须要有人才行，所以人是主表，而身份证是从表。设计从表可以有两种方案：
    在t_card表中添加外键列（相对t_user表），并且给外键添加唯一约束；
    给t_card表的主键添加外键约束（相对t_user表），即t_card表的主键也是外键。
一对多（多对一）：最为常见的就是一对多！一对多和多对一，这是从哪个角度去看得出来的。t_user和t_section的关系，从t_user来看就是一对多，而从t_section的角度来看就是多对一！这种情况都是在多方创建外键！
多对多：例如t_stu和t_teacher表，即一个学生可以有多个老师，而一个老师也可以有多个学生。这种情况通常需要创建中间表来处理多对多关系。例如再创建一张表t_stu_tea表，给出两个外键，一个相对t_stu表的外键，另一个相对t_teacher表的外键。
```



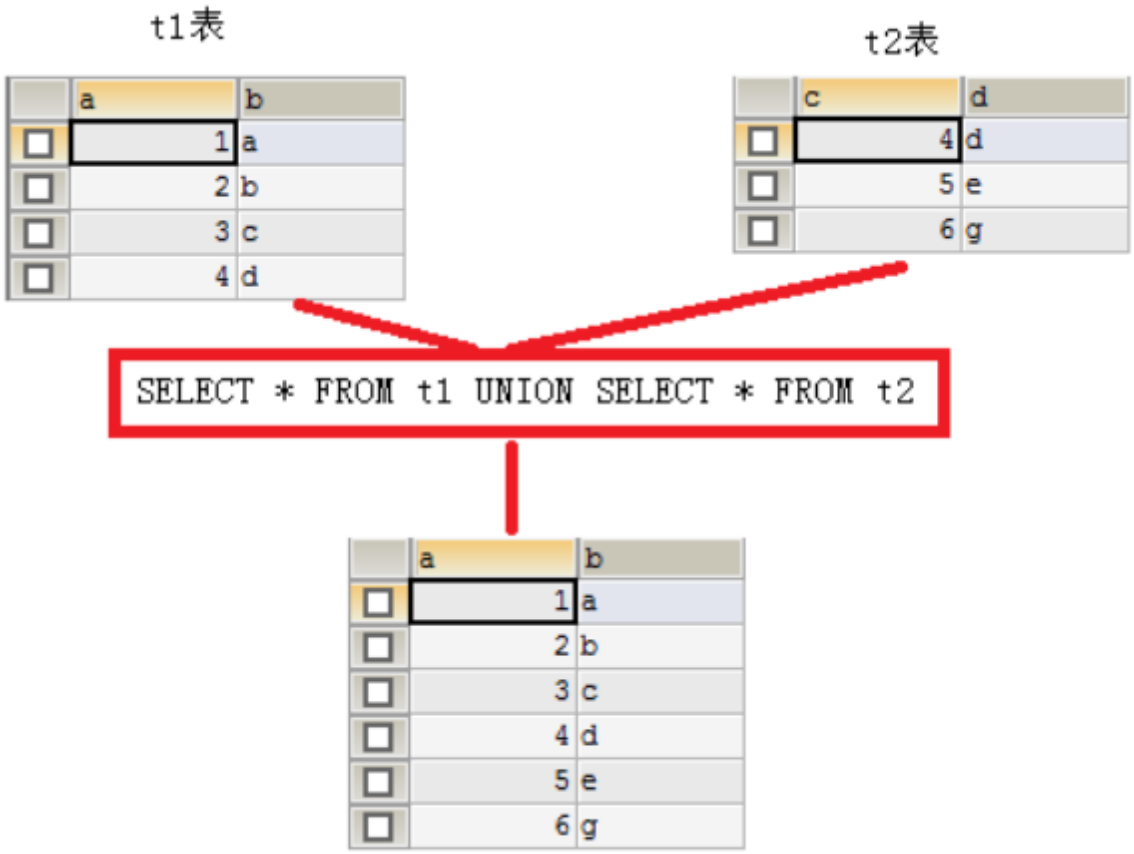
2. 多表查询

多表查询有如下几种:

```
合并结果集 UNION, UNION ALL
链接查询
    内连接 [INNER] JOIN ON
    外连接 OUTER JOIN ON
        左外连接 LEFT [OUTER] JOIN
        右外连接 RIGHT [OUTER] JOIN
        全外连接（MySQL不支持）FULL JOIN
    自然连接 NATURAL JOIN
子查询
```

2.1 合并结果集

1. 作用：合并结果集就是把两个select语句的查询结果合并到一起！
2. 合并结果集有两种方式：
   1)、UNION：去除重复记录，例如：SELECT * FROM t1 UNION SELECT * FROM t2；
   2)、 UNION ALL：不去除重复记录，例如：SELECT * FROM t1 UNION ALL SELECT * FROM t2。



2.2 连接查询

2.2.0 交叉连接

既然是多表查询，那么我们先来看看两张非常简单的表，我们就以这两张表为例，进行演示。

student表

sno	sname	sage	ssex	class_id
s001	张三	23	男	1
s002	李四	23	男	1
s003	吴鹏	25	男	2
s004	琴沁	20	女	1
s005	王丽	20	女	2
s006	李波	21	男	1
s007	刘玉	21	男	3
s008	萧蓉	21	女	1
s009	陈萧晓	23	女	2
s010	陈美	22	女	3

班级表

cid	cname
1	1班
2	2班
3	3班

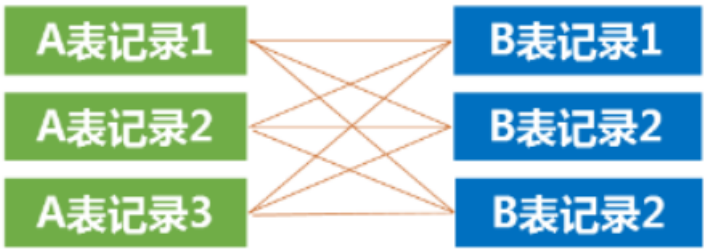
同时查询这两张表

```
select * from student,class;
```

信息 结果1 概况 状态

sno	sname	sage	ssex	class_id	cid	cname
s001	张三	23	男	1	1	1班
s001	张三	23	男	1	2	2班
s001	张三	23	男	1	3	3班
s002	李四	23	男	1	1	1班
s002	李四	23	男	1	2	2班
s002	李四	23	男	1	3	3班
s003	吴鹏	25	男	2	1	1班
s003	吴鹏	25	男	2	2	2班
s003	吴鹏	25	男	2	3	3班
s004	琴沁	20	女	1	1	1班
s004	琴沁	20	女	1	2	2班
s004	琴沁	20	女	1	3	3班
s005	王丽	20	女	2	1	1班
s005	王丽	20	女	2	2	2班
s005	王丽	20	女	2	3	3班
s006	李波	21	男	1	1	1班
s006	李波	21	男	1	2	2班
s006	李波	21	男	1	3	3班

仔细观察查询出的数据，可以发现，当使用上图中的语句时，student表中的每一行记录，都与class表中的任意一条记录相关联，同样，class表中的每一行记录，都与student表中的任意一条记录相关联。即当我们有2张表时,交叉连接的效果如下:



我们把上述"没有任何限制条件的连接方式"称之为"交叉连接"，"交叉连接"后得到的结果跟线性代数中的"笛卡尔乘积"一样。

```
笛卡尔积: 假设集合A={a,b}，集合B={0,1,2}，则两个集合的笛卡尔积为{(a,0),(a,1),(a,2),(b,0),(b,1),(b,2)}。可以扩展到多个集合的情况
```

```
SELECT * FROM student CROSS JOIN class;
```

只不过在mysql中 我们可以简化为上述的写法

多表查询产生这样的结果并不是我们想要的，那么怎么去除重复的，不想要的记录呢，当然是通过条件过滤。通常要查询的多个表之间都存在关联关系，那么就通过关联关系去除笛卡尔积

2.2.1 内连接 inner join

既然"交叉连接"不常用，那么肯定有其他的常用的"多表查询方式"。我们来看看另一种常用的多表查询的方式：内连接
扔用刚才的两张表student和class为例,首先回顾一下两张表的内容: student表

sno	sname	sage	ssex	class_id
s001	张三	23	男	1
s002	李四	23	男	1
s003	吴鹏	25	男	2
s004	琴沁	20	女	1
s005	王丽	20	女	2
s006	李波	21	男	1
s007	刘玉	21	男	3
s008	萧蓉	21	女	1
s009	陈萧晓	23	女	2
s010	陈美	22	女	3

班级表

cid	cname
1	1班
2	2班
3	3班

那么什么是"内连接"呢？我们可以把"内连接"理解成"两张表中同时符合某种条件的数据记录的组合"，例如在查出的数据中同时展现学生信息和他们所在的编号,示例如下:

```
SELECT * FROM student INNER JOIN class ON student.class_id = class.cid;
```

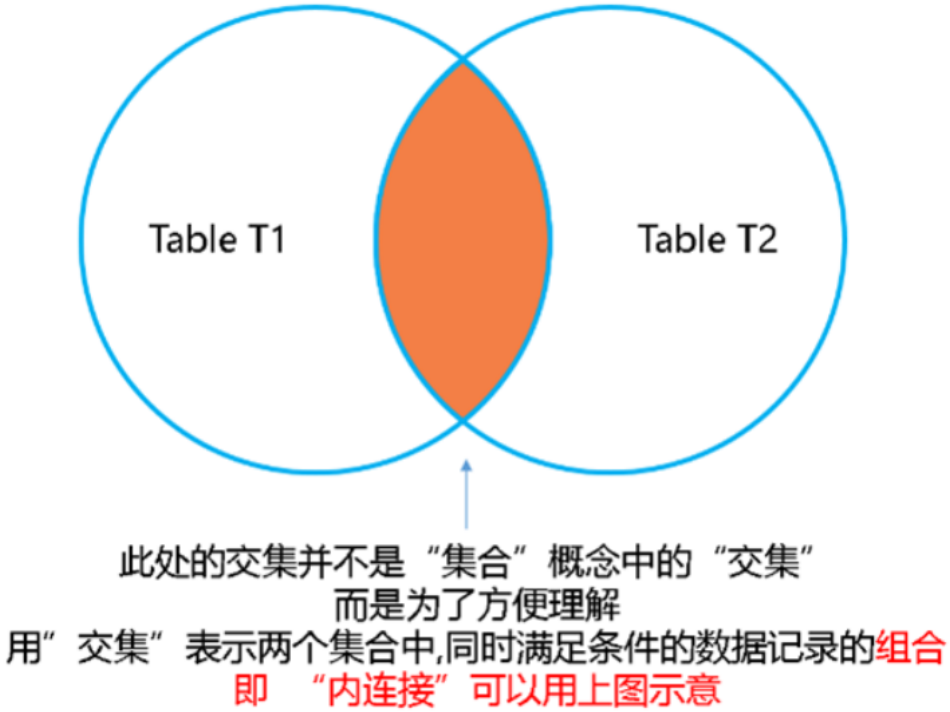
sno	sname	sage	ssex	class_id	cid	cname
s001	张三	23	男	1	1	1班
s002	李四	23	男	1	1	1班
s003	吴鹏	25	男	2	2	2班
s004	琴沁	20	女	1	1	1班
s005	王丽	20	女	2	2	2班
s006	李波	21	男	1	1	1班
s007	刘玉	21	男	3	3	3班
s008	萧蓉	21	女	1	1	1班
s009	陈萧晓	23	女	2	2	2班
s010	陈美	22	女	3	3	3班

上述的是等值内连接,还可以有不等值内连接

```
SELECT * FROM student INNER JOIN class ON student.class_id > class.cid;
```

sno	sname	sage	ssex	class_id	cid	cname
s003	吴鹏	25	男	2	1	1班
s005	王丽	20	女	2	1	1班
s007	刘玉	21	男	3	1	1班
s007	刘玉	21	男	3	2	2班
s009	陈萧晓	23	女	2	1	1班
s010	陈美	22	女	3	1	1班
s010	陈美	22	女	3	2	2班

这样内连接可以用图来表示:



使用"内连接"语句查询出的结果集是两个集合中"同时满足条件的数据"的"组合",所以我们并不能单纯的用"交集"去表示这个组合，就以上图为例，按照"交集"的定义，属于集合A且同时属于集合B的元素所组成的集合被称为交集，

2.2.2 外连接

外连接分为两种,左外连接,和右外连接,外连接的特点,结果集中有不满足情况的

左外连接

首先还是学生和班级表,不过我们学生表中插入了一个新学生,他还没有确定班级

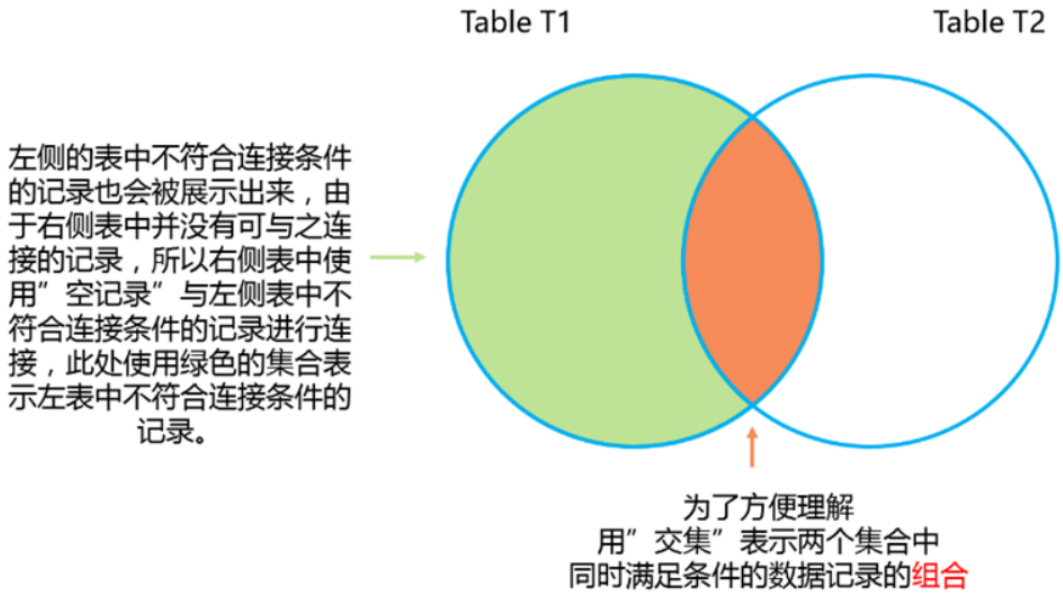
sno	sname	sage	ssex	class_id
s001	张三	23	男	1
s002	李四	23	男	1
s003	吴鹏	25	男	2
s004	琴沁	20	女	1
s005	王丽	20	女	2
s006	李波	21	男	1
s007	刘玉	21	男	3
s008	萧蓉	21	女	1
s009	陈萧晓	23	女	2
s010	陈美	22	女	(Null)

使用左外连接,查询学生信息和它对应的班级

```
SELECT * FROM student left JOIN class ON student.class_id = class.cid;
```

sno	sname	sage	ssex	class_id	cid	cname
s001	张三	23	男	1	1	1班
s002	李四	23	男	1	1	1班
s004	琴沁	20	女	1	1	1班
s006	李波	21	男	1	1	1班
s008	萧蓉	21	女	1	1	1班
s003	吴鹏	25	男	2	2	2班
s005	王丽	20	女	2	2	2班
s009	陈萧晓	23	女	2	2	2班
s007	刘玉	21	男	3	3	3班
s010	陈美	22	女	(Null)	(Null)	(Null)

左外连接不仅会查询出两表中同时符合条件的记录的组合，同时还会将"left join"左侧的表中的不符合条件的记录同时展示出来，由于左侧表中的这一部分记录并不符合连接条件，所以这一部分记录使用"空记录"进行连接。即左表的数据会完全的加载出来,即使右表中没有符合条件的数据



右连接

右连接与之类似

.....

2.2.3 子查询

一个select语句中包含另一个完整的select语句。
子查询就是嵌套查询，即SELECT中包含SELECT，如果一条语句中存在两个，或两个以上SELECT，那么就是子查询语句了。

```
子查询出现的位置
    where后，作为条为被查询的一条件的一部分
    from后，作表
当子查询出现在where后作为条件时，还可以使用如下关键字
    any
    all
子查询结果集的形式
    单行单列（用于条件）
    单行多列（用于条件）
    多行单列（用于条件）
    多行多列（用于表）
```

empno	ename	job	mgr	hiredate	sal	comm	deptno
7369	梁芷瑞	店员	7902	1980-12-17	800	(Null)	20
7499	杨鑫祥	售货员	7698	1981-02-20	1600	300	30
7521	陈泽名	售货员	7698	1981-02-22	1250	500	30
7566	吴世杰	经理	7839	1981-04-02	2975	(Null)	20
7654	李晓辰	售货员	7698	1981-09-28	1250	1400	30
7698	程晨强	经理	7839	1981-05-01	2850	(Null)	30
7782	张泳佳	经理	7839	1981-06-09	2450	(Null)	10
7788	彭钰盈	分析师	7566	1987-04-19	3000	(Null)	20
7839	刘添文	董事长	(Null)	1981-11-17	5000	(Null)	10
7844	林佳奇	售货员	7698	1981-09-08	1500	0	30
7876	黎铭辉	店员	7788	1987-05-23	1100	(Null)	20
7900	许旭濠	店员	7698	1981-12-03	950	(Null)	30
7902	廖佳林	分析师	7566	1981-12-03	3000	(Null)	20
7934	张辉	店员	7782	1982-01-23	1300	(Null)	10

工资高于张辉的员工

```sql
SELECT * FROM emp WHERE sal > (SELECT sal FROM emp WHERE ename='张辉')
-- 一行一列
```

查询与张泳佳同一个部门的员工

```sql
SELECT *
FROM lan_ou.emp
WHERE deptno = (SELECT deptno
                FROM lan_ou.emp
                WHERE ename = '张泳佳');
```

工资高于30号部门所有人的员工信息

```sql
SELECT * FROM emp WHERE sal>(
SELECT MAX(sal) FROM emp WHERE deptno=30);
```

查询有2个以上直接下属的员工信息

```sql
SELECT *
FROM emp
WHERE empno IN (
  SELECT mgr
  FROM emp
  GROUP BY mgr
  HAVING COUNT(mgr) >= 2);
```

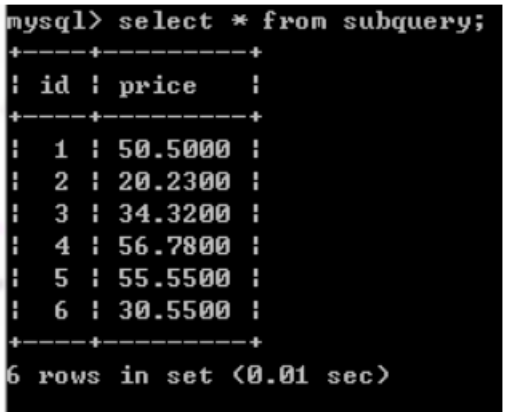
例如：

- 创建表

```sql
CREATE TABLE subquery (
    id tinyint(3) unsigned NOT NULL AUTO_INCREMENT,
    price float(6,4) NOT NULL,
    PRIMARY KEY (id)
)CHARSET=utf8;
```

插入数据

```sql
INSERT INTO subquery (price) VALUES
('50.5'),('20.23'),('34.32'),('56.78'),('55.55'),('30.55');
```



嵌套子查询

```sql
SELECT * FROM subquery WHERE price > (SELECT AVG(price) FROM subquery);
```

出现的问题：

```sql
SELECT * FROM subquery WHERE price > (SELECT price FROM subquery WHERE id >
3);
```

这个时候会报错， 因为子查询返回并不是一个值，而是一个集合值，所以这个时候我们就需要用到下面的ANY，SOME， ALL 或 IN；

用 ANY，SOME，ALL 修饰的比较运算符

- 关键字：
  ANY，SOME，ALL

例如：
还是同样的表，同样的数据

- 查询
  这里把上面报错的语句修改一下

```
SELECT * FROM subquery WHERE price > ANY(SELECT price FROM subquery WHERE id > 3);
```

```
mysql> SELECT * FROM subquery WHERE price > ANY(SELECT price FROM subquery WHERE
 id > 3);
+----+---------+
| id | price   |
+----+---------+
|  1 | 50.5000 |
|  3 | 34.3200 |
|  4 | 56.7800 |
|  5 | 55.5500 |
+----+---------+
4 rows in set (0.00 sec)
```

修饰的比较运算符的作用

	ANY	SOME	ALL
>, >=	最小值	最小值	最大值
<, <=	最大值	最大值	最小值
=	任意值	任意值	
!= , <>			任意值

解释： （ANY 修饰 >） 等价于 （查询出来的集合里只要外查询的字段值> 集合里的最小值）就满足情况。

测试：

```
SELECT price FROM subquery WHERE id > 3;
```

内部子查询结果：

```
mysql> SELECT price FROM subquery WHERE id > 3;
+---------+
| price   |
+---------+
| 56.7800 |
| 55.5500 |
| 30.5500 |
+---------+
3 rows in set (0.00 sec)
```

```
SELECT * FROM subquery WHERE price > ANY(SELECT price FROM subquery WHERE id > 3);
```

外部查询结果：

```
mysql> SELECT * FROM subquery WHERE price > ANY(SELECT price FROM subquery WHERE
 id > 3);
+----+---------+
| id | price   |
+----+---------+
|  1 | 50.5000 |
|  3 | 34.3200 |
|  4 | 56.7800 |
|  5 | 55.5500 |
+----+---------+
4 rows in set (0.00 sec)
```

如图所示，（ANY 修饰 >） 取出 子查询当中的最小值 30.5500， 表中只要price > (最小值 30.5500） 的行 满足情况。

使用 [NOT] IN 的子查询

- 关键字：
  IN， NOT IN
- 语法结构：
  operand comparison_operator [NOT] IN(subquery)
- **=ANY 与 IN 等效**
- **!=ALL 或 <>ALL 与 NOT IN 等效**

还是那张表，那些数据

```sql
SELECT * FROM subquery WHERE price IN (SELECT price FROM subquery WHERE id > 3);
```

```sql
SELECT * FROM subquery WHERE price = ANY (SELECT price FROM subquery WHERE id > 3);
```

```
mysql> SELECT * FROM subquery WHERE price IN (SELECT price FROM subquery WHERE i
d > 3);
+----+---------+
| id | price   |
+----+---------+
|  4 | 56.7800 |
|  5 | 55.5500 |
|  6 | 30.5500 |
+----+---------+
3 rows in set (0.00 sec)

mysql> SELECT * FROM subquery WHERE price = ANY(SELECT price FROM subquery WHERE
 id > 3);
+----+---------+
| id | price   |
+----+---------+
|  4 | 56.7800 |
|  5 | 55.5500 |
|  6 | 30.5500 |
+----+---------+
3 rows in set (0.00 sec)
```

```sql
SELECT * FROM subquery WHERE price NOT IN (SELECT price FROM subquery WHERE id > 3);
```

```sql
SELECT * FROM subquery WHERE price != ALL(SELECT price FROM subquery WHERE id > 3);
```

```
mysql> SELECT * FROM subquery WHERE price NOT IN (SELECT price FROM subquery WHE
RE id > 3);
+----+---------+
| id | price   |
+----+---------+
|  1 | 50.5000 |
|  2 | 20.2300 |
|  3 | 34.3200 |
+----+---------+
3 rows in set (0.00 sec)

mysql> SELECT * FROM subquery WHERE price != ALL(SELECT price FROM subquery WHER
E id > 3);
+----+---------+
| id | price   |
+----+---------+
|  1 | 50.5000 |
|  2 | 20.2300 |
|  3 | 34.3200 |
+----+---------+
3 rows in set (0.00 sec)
```

常用函数

数学函数

ABS(x) 返回x的绝对值
BIN(x) 返回x的二进制（OCT返回八进制，HEX返回十六进制）
CEILING(x) 返回大于x的最小整数值
EXP(x) 返回值e（自然对数的底）的x次方
FLOOR(x) 返回小于x的最大整数值
GREATEST(x1,x2,...,xn)返回集合中最大的值
LEAST(x1,x2,...,xn) 返回集合中最小的值
LN(x) 返回x的自然对数
LOG(x,y)返回x的以y为底的对数
MOD(x,y) 返回x/y的模（余数）
PI()返回pi的值（圆周率）
RAND()返回 0 到 1 内的随机值,可以通过提供一个参数(种子)使RAND()随机数生成器生成一个指定的值。
ROUND(x,y)返回参数x的四舍五入的有y位小数的值
SIGN(x) 返回代表数字x的符号的值
SQRT(x) 返回一个数的平方根
TRUNCATE(x,y) 返回数字x截短为y位小数的结果

日期函数

CURDATE()或CURRENT_DATE() 返回当前的日期
CURTIME()或CURRENT_TIME() 返回当前的时间
DATE_ADD(date,INTERVAL int keyword)返回日期date加上间隔时间int的结果(int必须按照关键字进行格式化),如：SELECTDATE_ADD- (CURRENT_DATE,INTERVAL 6 MONTH);
DATE_FORMAT(date,fmt) 依照指定的fmt格式格式化日期date值
DATE_SUB(date,INTERVAL int keyword)返回日期date加上间隔时间int的结果(int必须按照关键字进行格式化),如：SELECTDATE_SUB- (CURRENT_DATE,INTERVAL 6 MONTH);
DAYOFWEEK(date) 返回date所代表的一星期中的第几天(1~7)
DAYOFMONTH(date) 返回date是一个月的第几天(1~31)
DAYOFYEAR(date) 返回date是一年的第几天(1~366)

DAYNAME(date) 返回date的星期名，如：SELECT DAYNAME(CURRENT_DATE);
FROM_UNIXTIME(ts,fmt) 根据指定的fmt格式，格式化UNIX时间戳ts
HOUR(time) 返回time的小时值(0~23)
MINUTE(time) 返回time的分钟值(0~59)
MONTH(date) 返回date的月份值(1~12)
MONTHNAME(date) 返回date的月份名，如：SELECT MONTHNAME(CURRENT_DATE);
NOW() 返回当前的日期和时间
QUARTER(date) 返回date在一年中的季度(1~4)，如SELECT QUARTER(CURRENT_DATE);
WEEK(date) 返回日期date为一年中第几周(0~53)
YEAR(date) 返回日期date的年份(1000~9999)
LAST_DAY(date) 需要一个日期或日期时间值，并返回该月的最后一天对应的值。 如果该参数是无效的，则返回NULL。

字符串类

ASCII(char)返回字符的ASCII码值
BIT_LENGTH(str)返回字符串的比特长度
CONCAT(s1,s2...,sn)将s1,s2...,sn连接成字符串
CONCAT_WS(sep,s1,s2...,sn)将s1,s2...,sn连接成字符串，并用sep字符间隔
INSERT(str,x,y,instr) 将字符串str从第x位置开始，y个字符长的子串替换为字符串instr，返回结果
FIND_IN_SET(str,list)分析逗号分隔的list列表，如果发现str，返回str在list中的位置
LCASE(str)或LOWER(str) 返回将字符串str中所有字符改变为小写后的结果
LEFT(str,x)返回字符串str中最左边的x个字符
LENGTH(s)返回字符串str中的字符数
LTRIM(str) 从字符串str中切掉开头的空格
POSITION(substr,str) 返回子串substr在字符串str中第一次出现的位置
QUOTE(str) 用反斜杠转义str中的单引号
REPEAT(str,srchstr,rplcstr)返回字符串str重复x次的结果
REVERSE(str) 返回颠倒字符串str的结果
RIGHT(str,x) 返回字符串str中最右边的x个字符
RTRIM(str) 返回字符串str尾部的空格
STRCMP(s1,s2)比较字符串s1和s2
TRIM(str)去除字符串首部和尾部的所有空格
UCASE(str)或UPPER(str) 返回将字符串str中所有字符转变为大写后的结果

mysqldump –hlocalhost –uroot –p databaseName >d:\dump.sql (window)

sql语句练习

```
1、查询“c001”课程比“c002”课程成绩高的所有学生的学号；

select a.* from
 (select * from sc a where a.cno='c001') a,
 (select * from sc b where b.cno='c002') b
 where a.sno=b.sno and a.score > b.score;

或

select * from sc a
 where a.cno='c001'
```

```
 and  exists(select * from sc b where b.cno='c002' and a.score>b.score and a.sno = b.sno)

*************************************************************

2、查询平均成绩大于60 分的同学的学号和平均成绩;

select sno,avg(score) from sc  group by sno having avg(score)>60;

*************************************************************

3、查询所有同学的学号、姓名、选课数、总成绩;

select a.*,s.sname from (select sno,sum(score),count(cno) from sc group by sno) a ,student s where a.sno=s.sno

*************************************************************

4、查询姓“刘”的老师的个数;

select count(*) from teacher where tname like '刘%';

*************************************************************

5、查询没学过“谌燕”老师课的同学的学号、姓名;

select a.sno,a.sname from student a where a.sno not in
(
 select distinct s.sno from sc s,
      (
       select c.* from course c ,
           (select tno from teacher t where tname='谌燕') t
       where c.tno=t.tno
      ) b
  where s.cno = b.cno
)
或
select * from student st where st.sno not in
(
 select distinct sno from sc s
 join course c on s.cno=c.cno
 join teacher t on c.tno=t.tno where tname='谌燕'
)

*************************************************************

6、查询学过“c001”并且也学过编号“c002”课程的同学的学号、姓名;

select st.* from sc a
 join sc b on a.sno=b.sno
 join student st on st.sno=a.sno
 where a.cno='c001' and b.cno='c002' and st.sno=a.sno;

*************************************************************

7、查询学过“谌燕”老师所教的所有课的同学的学号、姓名;

select st.* from student st join sc s on st.sno=s.sno
```

```
 join course c on s.cno=c.cno
 join teacher t on c.tno=t.tno
 where t.tname='谌燕'

*************************************************************

8、查询课程编号“c002”的成绩比课程编号“c001”课程低的所有同学的学号、姓名；

select * from student st
 join sc a on st.sno=a.sno
 join sc b on st.sno=b.sno
 where a.cno='c002' and b.cno='c001' and a.score < b.score
*************************************************************

9、查询所有课程成绩小于60 分的同学的学号、姓名；

select st.*,s.score from student st
 join sc s on st.sno=s.sno
 join course c on s.cno=c.cno
 where s.score <60

*************************************************************

10、查询没有学全所有课的同学的学号、姓名；

select stu.sno,stu.sname,count(sc.cno) from student stu
 left join sc on stu.sno=sc.sno
 group by stu.sno,stu.sname
 having count(sc.cno)<(select count(distinct cno)from course)

或
select * from student where sno in
(
 select sno from
        (
         select stu.sno,c.cno from student stu cross join course c minus
select sno,cno from sc
        )
)

*************************************************************

11、查询至少有一门课与学号为“s001”的同学所学相同的同学的学号和姓名；

select st.* from student st,
 (select distinct a.sno from
  (select * from sc) a,
  (select * from sc where sc.sno='s001') b
  where a.cno=b.cno
 ) h
 where st.sno=h.sno and st.sno<>'s001'

*************************************************************

12、查询至少学过学号为“s001”同学所有一门课的其他同学学号和姓名；

select * from sc
 left join student st on st.sno=sc.sno
```

```
 where sc.sno<>'s001'
 and sc.cno in (select cno from sc where sno='s001')

*************************************************************

13、把“SC”表中“谌燕”老师教的课的成绩都更改为此课程的平均成绩；

update sc c set score=
 (
  select avg(c.score)  from course a,teacher b
  where a.tno=b.tno and b.tname='谌燕' and a.cno=c.cno
  group by c.cno
 )
where cno in( select cno from course a,teacher b where a.tno=b.tno and
b.tname='谌燕')

*************************************************************

14、查询和“s001”号的同学学习的课程完全相同的其他同学学号和姓名；

select* from sc where sno<>'s001'
minus
( select* from sc minus select * from sc where sno='s001')

*************************************************************

15、删除学习“谌燕”老师课的SC 表记录；

delete from sc where sc.cno in
(
 select cno from course c
 left join teacher t on  c.tno=t.tno
 where t.tname='谌燕'
)

*************************************************************

16、向SC 表中插入一些记录，这些记录要求符合以下条件：没有上过编号“c002”课程的同学学
号、“c002”号课的平均成绩；

insert into sc (sno,cno,score)
 select distinct st.sno,sc.cno,(select avg(score)from sc where cno='c002')
 from student st,sc
 where not exists
 (select * from sc where cno='c002' and sc.sno=st.sno) and sc.cno='c002';

*************************************************************

17、查询各科成绩最高和最低的分：以如下形式显示：课程ID，最高分，最低分

select cno ,max(score),min(score) from sc group by cno;

*************************************************************

18、按各科平均成绩从低到高和及格率的百分数从高到低顺序

select cno,avg(score),sum(case when score>=60 then 1 else 0 end)/count(*)
as 及格率
```

```
from sc group by cno
order by avg(score) , 及格率 desc

**********************************************************************

19、查询不同老师所教不同课程平均分从高到低显示

select max(t.tno),max(t.tname),max(c.cno),max(c.cname),c.cno,avg(score)
from sc , course c,teacher t
where sc.cno=c.cno and c.tno=t.tno
group by c.cno
order by avg(score) desc

**********************************************************************

20、统计列印各科成绩,各分数段人数:课程ID,课程名称,[100-85],[85-70],[70-60],[ <60]

select sc.cno,c.cname,
sum(case  when score between 85 and 100 then 1 else 0 end) AS "[100-85]",
sum(case  when score between 70 and 85 then 1 else 0 end) AS "[85-70]",
sum(case  when score between 60 and 70 then 1 else 0 end) AS "[70-60]",
sum(case  when score <60 then 1 else 0 end) AS "[<60]"
from sc, course c
where  sc.cno=c.cno
group by sc.cno ,c.cname;

**********************************************************************

21、查询各科成绩前三名的记录:(不考虑成绩并列情况)

select * from
(select sno,cno,score,row_number()over(partition by cno order by score
desc) rn from sc)
where rn<4

**********************************************************************

22、查询每门课程被选修的学生数

select cno,count(sno)from sc group by cno;

**********************************************************************

23、查询出只选修了一门课程的全部学生的学号和姓名

select sc.sno,st.sname,count(cno) from student st
left join sc
on sc.sno=st.sno
group by st.sname,sc.sno having count(cno)=1;

**********************************************************************

24、查询男生、女生人数

select ssex,count(*)from student group by ssex;

**********************************************************************
```

```
25、查询姓“张”的学生名单

select * from student where sname like '张%';

**********************************************************************

26、查询同名同性学生名单，并统计同名人数

select sname,count(*)from student group by sname having count(*)>1;

**********************************************************************

27、1981 年出生的学生名单(注：Student 表中Sage 列的类型是number)

select sno,sname,sage,ssex from student t where to_char(sysdate,'yyyy')-
sage =1988

**********************************************************************

28、查询每门课程的平均成绩，结果按平均成绩升序排列，平均成绩相同时，按课程号降序排列

select cno,avg(score) from sc group by cno order by avg(score)asc,cno desc;

**********************************************************************

29、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 st.sno,st.sname,avg(score) from student st
left join sc on sc.sno=st.sno
group by st.sno,st.sname having avg(score)>85;

**********************************************************************

30、查询课程名称为“数据库”，且分数低于60 的学生姓名和分数

select sname,score from student st,sc,course c
where st.sno=sc.sno and sc.cno=c.cno and c.cname='Oracle' and sc.score<60

**********************************************************************

31、查询所有学生的选课情况；

select st.sno,st.sname,c.cname from student st,sc,course c
where sc.sno=st.sno and sc.cno=c.cno;

**********************************************************************

32、查询任何一门课程成绩在70 分以上的姓名、课程名称和分数；

select st.sname,c.cname,sc.score from student st,sc,course c
where sc.sno=st.sno and sc.cno=c.cno and sc.score>70

**********************************************************************

33、查询不及格的课程，并按课程号从大到小排列

select sc.sno,c.cname,sc.score from sc,course c
where sc.cno=c.cno and sc.score<60 order by sc.cno desc;
```

```
```

```
**************************************************************
```

34、查询课程编号为c001 且课程成绩在80 分以上的学生的学号和姓名；

```
select st.sno,st.sname,sc.score from sc,student st
where sc.sno=st.sno and cno='c001' and score>80;
```

```
**************************************************************
```

35、求选了课程的学生人数

```
select count(distinct sno) from sc;
```

```
**************************************************************
```

36、查询选修“谌燕”老师所授课程的学生中，成绩最高的学生姓名及其成绩

```
select st.sname,score from student st,sc ,course c,teacher t
where
st.sno=sc.sno and sc.cno=c.cno and c.tno=t.tno and t.tname='谌燕' and
sc.score=
(select max(score)from sc where sc.cno=c.cno)
```

```
**************************************************************
```

37、查询各个课程及相应的选修人数

```
select cno,count(sno) from sc group by cno;
```

```
**************************************************************
```

38、查询不同课程成绩相同的学生的学号、课程号、学生成绩

```
select a.* from sc a ,sc b where a.score=b.score and a.cno<>b.cno
```

```
**************************************************************
```

39、查询每门功课成绩最好的前两名

```
select * from (
 select sno,cno,score,row_number()over(partition by cno order by score
desc) my_rn from sc t
)
where my_rn<=2
```

```
**************************************************************
```

40、统计每门课程的学生选修人数（超过10 人的课程才统计）。要求输出课程号和选修人数，查询结果按人数降序排列，若人数相同，按课程号升序排列

```
select cno,count(sno) from sc
group by cno having count(sno)>10
order by count(sno) desc,cno asc;
```

```
**************************************************************
```

41、检索至少选修两门课程的学生学号

```
select sno from sc group by sno having count(cno)>1;
或
select sno from sc group by sno having count(sno)>1;

*************************************************************

42、查询全部学生都选修的课程的课程号和课程名

select distinct(c.cno),c.cname from course c ,sc where sc.cno=c.cno
或
select cno,cname from course c where c.cno in (select cno from sc group by
cno)

*************************************************************

43、查询没学过“谌燕”老师讲授的任一门课程的学生姓名

select st.sname from student st
where st.sno not in
(select distinct sc.sno from sc,course c,teacher t where sc.cno=c.cno and
c.tno=t.tno and t.tname='谌燕')

*************************************************************

44、查询两门以上不及格课程的同学的学号及其平均成绩

select sno,avg(score)from sc
where sno in
(
 select sno from sc where sc.score<60
 group by sno having count(sno)>1
) group by sno

*************************************************************

45、检索“c004”课程分数小于60，按分数降序排列的同学学号

select sno from sc where cno='c004' and score<90 order by score desc;

*************************************************************

46、删除“s002”同学的“c001”课程的成绩

delete from sc where sno='s002' and cno='c001';
```