上一次课：数据的增删改查（简单的查询）

查询的语法格式：

```
select *|字段 1，字段 2，…… --5
from 表名     --1
[where 查询条件] --2
[group by 分组字段] --3
[having 过滤条件] --4
[order by 字段 asc|desc]; --6
```

注解：

select 子句：过滤出满足条件的列（字段）
from 子句：确认数据来自哪些表
where 子句：过滤出满足条件的行(分组之前的过滤）
group by 子句：对数据做分组
having 子句：对分组之后的语句做进一步的过滤
order by 子句：对查询出来的数据进行排序显示

分组之后的进一步过滤 having：

示例：

查询有两门及其以上课程不及格的学生学号

思考 1：

如何查询课程不及格的学生学号？

```
use myforthdb;
select xh
from cjb
where cj<60;
```

思考 2：

如何在 1 的基础上加上限定条件两门及其以上呢？——》
按学号进行分组，统计各个学号的个数，过滤个数>1 的记录

```
use myforthdb;
select xh
```

```
from cjb
where cj<60
group by xh
having count(*)>1;
```

注意：

1）where 和 having 后面跟的都是过滤条件，但是 where 是分组之前的过滤，having 是分组之后的过滤，所以，聚合函数（统计函数）不会出现在 where 子句里。
2）代码的错误分两类：语法错误；逻辑错误

删除：

删除某一个数据：update xx set y=null where z=10;
删除某一列数据（包含表头）：alter table xx drop 字段名;
删除某一列数据（不包含表头）：update xx set y=null;
删除所有的行（保留表结构）：delete from xx;
删除所有的行（不保留表结构）：drop table xx;
删除特定的行：delete from xx where ...;

```
create table stu(
    sid int,
    name char(20)
);

insert into stu(sid,name) values(1,'zs');
insert into stu values(2,'lis');   -- 正确
insert into stu values('ww',3);    -- 错误，字段省略的插入方式，字段值要按建表的顺序来写
```

插入一条记录（名字为空）

```
insert into stu(sid,name) values(4,null);
insert into stu(sid) values(5);

select * from stu;
```

空值查询：

```
select * from stu where name is null;
select * from stu where name=null; -- 错误写法

name like 'hgf_t%'
```

name like 'hgf%' -- 范围最大
name like 'hgf_'

多表查询

1）子查询——》查询嵌套——》查询里面嵌套了查询

实例 1：查询选修了 java 语言课的全部学生的平均成绩

分析：题目里涉及的数据来自于哪些表？——》java 语言是课程名，来自于 kcb，成绩来自于 cjb。

我们找关联关系：

（1）首先可以在课程表里根据 java 语言找到对应的课程号；
select kch from kcb where kcm='java 语言';
（2）根据第一步查到的课程号在成绩表里找到对应的成绩并求平均值
select avg(cj) from cjb where kch=(select kch from kcb where kcm='java 语言');

select round(avg(cj)) from cjb where kch=(select kch from kcb where kcm='java 语言');
根据 kcb 找到课程号 02,然后再根据 cjb 的课程名去找到对应的学员信息,然后在求出对应的学员平均成绩
Round 为保留整数位

示例 2：查询 1 班叫张三的同学的 java 语言的成绩

分析：1 班、张三来自于学生表；java 语言来自于课程表，成绩来自于成绩表
——》如果要查成绩，那么需要确定该成绩所对应的学号和课程号——》如何确定学号和课程号呢？
——》根据 1 班张三信息可以在学生表里确定学号，根据 java 语言课程名可以在课程表里确定课程号

（1）确定 1 班张三所对应的学号：

select xh from xsb where bj='1 班' and xm='张三';

（2）确定 java 语言所对应的课程号：

select kch from kcb where kcm='java 语言';

（3）在成绩表里查询该学号对应的成绩：

select cj from cjb where xh=(select xh from xsb where bj='1 班' and xm='张三')
and kch=(select kch from kcb where kcm='java 语言');

---

2)关联查询——》将多张表（两张及其以上的表）联接成一张大的表，然后从表中查询特定的数据

引入：查询张三同学的各科成绩，要求显示的字段有姓名、课程号和成绩

分析：姓名来自于学生表，课程号和成绩来自于成绩表——》我们现在的查询需求是希望在查询结果里，
不在一张表的字段信息也能在一张表里显示——》关联查询可以满足这个需求。

表和表之间如何做关联（合并）呢？——》笛卡儿积运算——》什么是笛卡儿积运算呢？将两张表里任意两个记录（两行）组合在一起形成新的记录，最终形成一张大的表的过程。

例如：学生表和成绩表做笛卡儿积运算:

```
select * from xsb,cjb;
```

——》运算后生成的大的表：132 行（11*12）11 列（8+3）

笛卡儿积运算虽然将两张表关联成了一张大的表，但是这张大的表里存在许多没有意义的垃圾数据
（例如张三学员信息会关联不是张三的学员的成绩，这样的数据是没有意义的）——》
既然笛卡儿积运算会产生垃圾数据，如何过滤垃圾数据？——》可以通过添加关联条件的方式进行过滤
——》如何寻找关联条件呢？——》我们希望张三的信息连接张三的各科成绩，
所以我们可以通过让学生表的学号跟成绩表的学号相等这个条件来过滤垃圾数据
——》select * from xsb,cjb where xsb.xh=cjb.xh;

回到最初始的查询需求：

查询张三同学的各科成绩，要求显示的字段有姓名、课程号和成绩

```
select xm,kch,cj
from xsb,cjb
where xsb.xh=cjb.xh
and xm='张三';
```

注解：

（1）from 子句后面可以跟多张表，多张表之间用英文的逗号分隔，
只要看到 from 后面跟两张或以上的表，就要反应出这是在做笛卡儿积运算，
笛卡儿积运算会产生垃圾数据，需要添加关联条件来消除产生的垃圾数据；
（2）如果题目的需求除了多表关联以外还有其他的限定条件，我们通过 and 连接其他的条件；
（3）在关联查询中，涉及到同名的字段，一定要加上表名做前缀，
否则数据库无法识别该字段数据哪张表，从而会报错。

对于初学者来说，可以选择在关联查询里将所有的字段都带上表头：

```
select xsb.xm,cjb.kch,cjb.cj
from xsb,cjb
where xsb.xh=cjb.xh
and xsb.xm='张三';
```

(4)除了可以给字段取别名之外，还可以给表取别名,也就是缩写了查询条件,也可以查询出来:

```
select x.xm,c.kch,c.cj
from xsb x,cjb c
where x.xh=c.xh
and x.xm='张三';
```

(5)from a,b——》做一次笛卡儿积运算，
from a,b,c——》做两次笛卡儿积运算，a 和 b 做完笛卡儿积运算生成一张大的表，生成的
大的表再跟 c 做笛卡儿积运算，最终生成一张更大的表
也就是说先两个运算,然后大表再跟第三张表运算

扩展:

关联查询有多种：

内联接：

只返回满足关联条件的结果集 ——上面的例子就是内联接 inner join

外联接：

（1）左外联接：left join

指的是除了返回满足关联条件的结果集以外，还会把左边的那张表完整的展示出来，
右边那张表里不满足关联条件的字段位置补空值（null)

（2）右外联接：right join

指的是除了返回满足关联条件的结果集以外，还会把右边的那张表完整的展示出来，
左边那张表里不满足关联条件的字段位置补空值（null)

（3）全外联接：full join

指的是除了返回满足关联条件的结果集以外，还会把两边的表完整的展示出来，
两边表里不满足关联条件的字段位置补空值（null)

外联接的示例：

查询所有学生的课程号、学号、姓名、成绩，要求没有成绩信息的学员信息也要查询出来。

不使用外链接:

```
select cjb.kch,cjb.xh,xsb.xm,cjb.cj
from xsb,cjb
where xsb.xh=cjb.xh;
```

使用内链接:

```
select cjb.kch,cjb.xh,xsb.xm,cjb.cj
from xsb inner join cjb
on xsb.xh=cjb.xh;
```

最后把 where 还为 on

以上查询语句不能将没有成绩的学员信息也查询出来，
如果想将没有成绩的学员信息查询出来，需要使用外联接（左外联接和右外联接都可以实现）

```
select cjb.kch,cjb.xh,xsb.xm,cjb.cj
from xsb left join cjb
on xsb.xh=cjb.xh;

select cjb.kch,cjb.xh,xsb.xm,cjb.cj
from cjb right join xsb
on xsb.xh=cjb.xh;
```

全外联接：全外联接通常使用 full join 来实现。
然而，MySQL 不支持 full join,但是在别的数据库（Oracle）中是支持的。
那么在 MySQL 里可以如何实现全外联接呢？
——》可以通过 union 集合操作来实现：

以下查询会将两张表（xsb 和 cjb）里的记录完整的展示出来：

```
select cjb.kch,cjb.xh,xsb.xm,cjb.cj
from xsb left join cjb
on xsb.xh=cjb.xh
union
select cjb.kch,cjb.xh,xsb.xm,cjb.cj
from cjb left join xsb
on xsb.xh=cjb.xh;
```