MySQL（五）

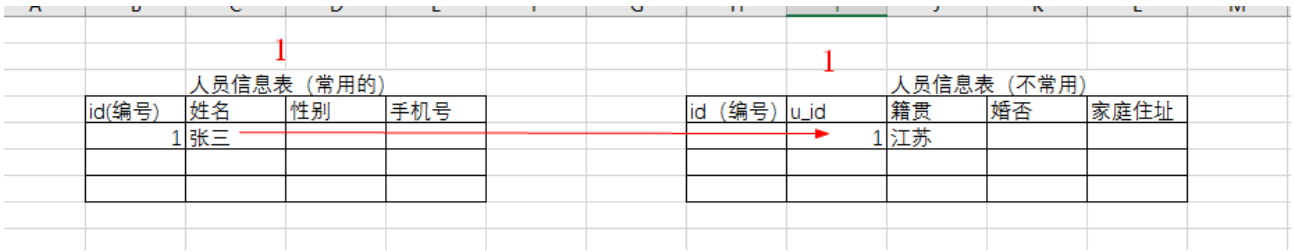
1. 多表关系

在 MySQL 数据库中表和表之间存在一些必要的关系，他们分别是：一对一、一对多、多对多关系，下面我们来具体学习一下。

1.1 一对一关系

一对一：一张表中的一条记录与另外一张表中最多有一条明确的对应关系

如何设置一对一：
在另外一个表《人员信息表（不常用）》中增加一个字段，设置字段的值和另外一表《人员信息表》中的
id 相等，而且在《人员信息表（不常用）》中有且只有一条数据对应



应用场合：
把一个很复杂的表，拆分为多个表，会用一对一的关系

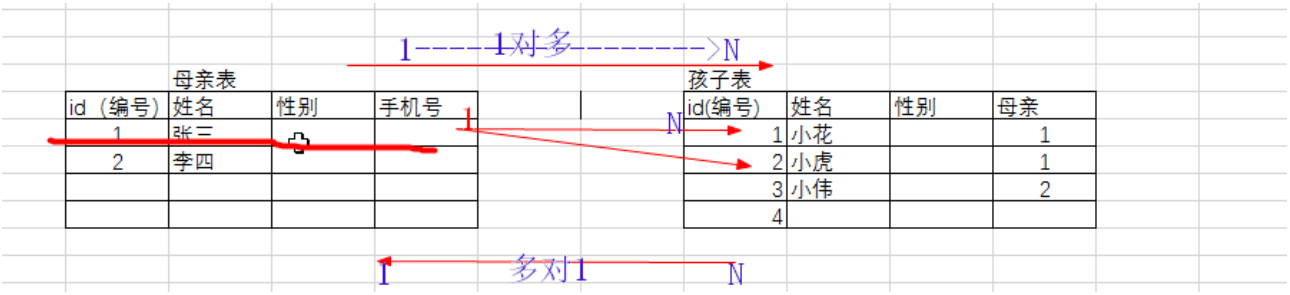
```
-- 创建 person 表（常用表）
create table person(
    id int(10) unsigned primary key auto_increment,
    name varchar(10) not null,
    sex enum("男","女"),
    telnum bigint(11)
);
-- 插入数据到 person 表中
insert into person values(null,'张三','男',13800138000);
insert into person values(null,'李四','男',13800138000);
insert into person values(null,'王五','男',13800138000);

-- 创建人员信息详情表（不常用）
```

```
create table person_detail(
    id int(10) unsigned primary key auto_increment,
    u_id int(10) unsigned not null,
    jiguan varchar(10) not null,
    is_merry_state enum("y","n"),
    address varchar(200)
);

-- 添加人员详细数据
insert into person_detail values(null,1,'山东','y','山东省济南市');
insert into person_detail values(null,2,'河南','n','河南省郑州市');
insert into person_detail values(null,3,'河北','y','河北省石家庄市');
```

1.2 一对多关系

一对多：在一个表中的一条数据和另外一个表中的多条数据形成关联关系，他就是一对多的关系，反过来当一个表中多条数据和另外一个表中的一条数据对应，这就是多对一的关系

如何建立一对多的关系
1）在“多”的表中，增加一个字段，这个字段可以和“一”表中的编号相等，“多”表中可以有多条记录和“一”表中的编号相等

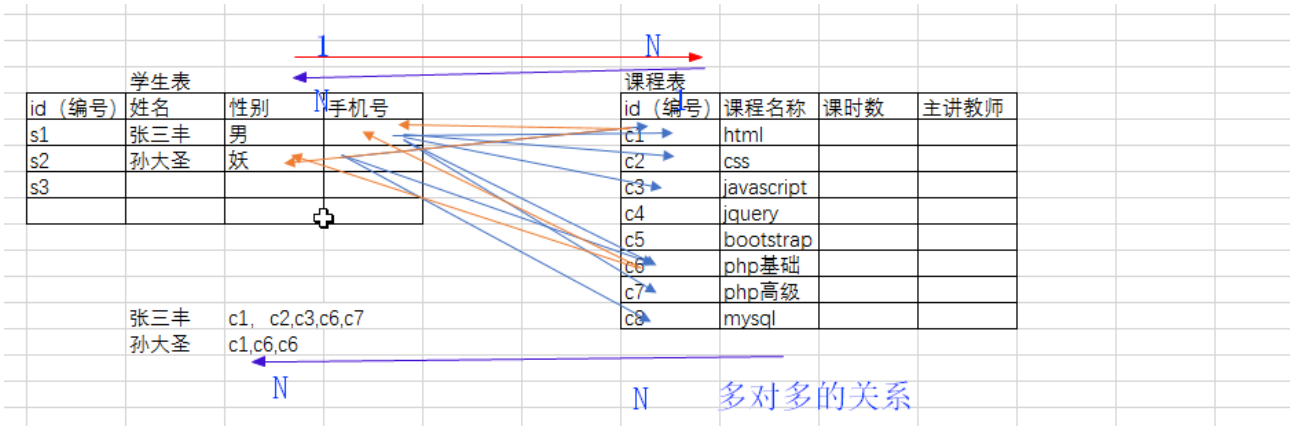



```
-- 创建母亲表
create table mother(
    id int(10) unsigned primary key auto_increment,
    name varchar(10) not null,
    telnum bigint(11)
);

insert into mother values(null,'兰兰',13800138000);
insert into mother values(null,'花花',13800138000);
insert into mother values(null,'美美',13800138000);

-- 创建孩子表
```

```
create table children(
    id int(10) unsigned primary key auto_increment,
    name varchar(10) not null,
    sex enum("男","女"),
    mother_id int(10) unsigned not null;
);
insert into children values(null,'可可','男');
insert into children values(null,'小米粒','女');
insert into children values(null,'小丸子','女');
```

1.3 多对多关系

多对多：一张表中的一条记录在另外一张表中可以匹配到多条记录，反过来也一样。



多对多的关系如果按照多对一的关系维护：就会出现一个字段中有多个其他表的主键，在访问的时候就会带来不便。

既然通过两张表自己增加字段解决不了问题，那么就通过第三张表来解决。

师生关系

1、 一个学生学习多个课程；

2、 一个课程被多个学生选课学习；

首先得有两个实体：学生表和课程表

学生表

id（编号）	姓名	性别	手机号
s1	张三丰	男	
s2	孙大圣	妖	
s3			

课程表

id（编号）	课程名称	课时数	主讲教师
c1	html		
c2	css		
c3	javascript		
c4	jquery		
c5	bootstrap		
c6	php基础		
c7	php高级		
c8	mysql		

张三丰 c1，c2,c3,c6,c7
孙大圣 c1,c6,c8

从中间设计一张表：维护两张表对应的联系：每一种联系都包含

学生表

id（编号）	姓名	性别	手机号
s1	张三丰	男	
s2	孙大圣	妖	
s3			

课程表

id（编号）	课程名称	课时数	主讲教师
c1	html		
c2	css		
c3	javascript		
c4	jquery		
c5	bootstrap		
c6	php基础		
c7	php高级		
c8	mysql		

张三丰 c1，c2,c3,c6,c7
孙大圣 c1,c6,c8

维护多对多的关系，通常情况下是新建一张表
学生-课程表

id（编号）	学生编号	课程编号
	s1	c1
	s1	c2
	s1	c3
	s1	c6
	s1	c7
	s2	c1
	s2	c6
	s2	c8

多对多解决方案；增加一个中间表，让中间表与对应的其他表形成两个多对一的关系：多对一的解决方案是在“多”表中增加“一”表对应的主键字段。


```
# 学生表
create table students
(
    id int unsigned not null auto_increment comment '编号',
    name varchar(20) not null comment '姓名',
    gender enum('男','女') not null comment '性别',
    tel bigint unsigned not null comment '手机号',
    primary key(id)
) comment='学生表';

    insert into students
    values
    (1,'八戒','男',13912345555),
    (2,'大师兄','男',13912346666),
    (3,'金角大王','男',13922222222);

# 课程表
create table course
(
    id int unsigned not null auto_increment comment '编号',
    course_name varchar(50) not null comment '课程名称',
    primary key(id)
) comment='课程表';
```

```
        insert into course
        values(1,'HTML'),(2,'CSS'),(3,'JavaScript'),
        (4,'jQuery'),(5,'Bootstrap'),(6,'PHP'),(7,'MySQL');


    # 学生选课表
    create table student_course
    (
        student_id int unsigned not null comment '学生 Id',
        course_id int unsigned not null comment '课程 Id'
    ) comment='学生选课表';

        # id=1,八戒         学习了      PHP、MySQL
        # id=2,大师兄       学习了      HTML、CSS、JavaScript、jQuery、Bootstrap、
PHP、MySQL
        # id=3,金角大五      学习了      HTML、CSS、JavaScript
    insert into student_course
    values
        (1,6),(1,7),
        (2,1),(2,2),(2,3),(2,4),(2,5),(2,6),(2,7),
        (3,1),(3,2),(3,3);
```

2. 联合查询

2.1 基本概念

联合查询是可合并多个相似的 select 查询的结果集。等同于**将一个表追加到另一个表**，从而实现将两个表的查询组合到一起，使用谓词为 UNION 或 UNION ALL。

联合查询：将多个查询的结果合并到一起（纵向合并）：字段数不变，多个查询的记录数合并。

2.2 应用场景

1、 将同一张表中不同的结果（需要对应多条查询语句来实现），合并到一起展示数据
男生身高升序排序，女生身高降序排序

2、 最常见：在数据量大的情况下，会对表进行分表操作，需要对每张表进行部分数据统计，

使用联合查询来讲数据存放到一起显示。

2.3 基本语法

基本语法：
Select 语句
Union [union 选项]
Select 语句;

Union 选项：与 select 选项基本一样
Distinct：去重，去掉完全重复的数据（默认的）

```
mysql> select * from person;
+----+------+------+-------------+
| id | name | sex  | telnum      |
+----+------+------+-------------+
|  1 | 张三 | 男   | 13800138000 |
|  2 | 李四 | 男   | 13800138000 |
|  3 | 王五 | 男   | 13800138000 |
+----+------+------+-------------+
3 rows in set (0.00 sec)

mysql> select * from student;
ERROR 1146 (42S02): Table 'czxy.student' doesn't exist
mysql> select * from students;
+----+----------+--------+-------------+
| id | name     | gender | tel         |
+----+----------+--------+-------------+
|  1 | 八戒     | 男     | 13912345555 |
|  2 | 大师兄   | 男     | 13912346666 |
|  3 | 金角大王 | 男     | 13922222222 |
+----+----------+--------+-------------+
3 rows in set (0.00 sec)
```

```
mysql> select * from person
    -> union
    -> select * from students;
+----+----------+------+-------------+
| id | name     | sex  | telnum      |
+----+----------+------+-------------+
|  1 | 张三     | 男   | 13800138000 |
|  2 | 李四     | 男   | 13800138000 |
|  3 | 王五     | 男   | 13800138000 |
|  1 | 八戒     | 男   | 13912345555 |
|  2 | 大师兄   | 男   | 13912346666 |
|  3 | 金角大王 | 男   | 13922222222 |
+----+----------+------+-------------+
6 rows in set (0.00 sec)
```

All：保存所有的结果

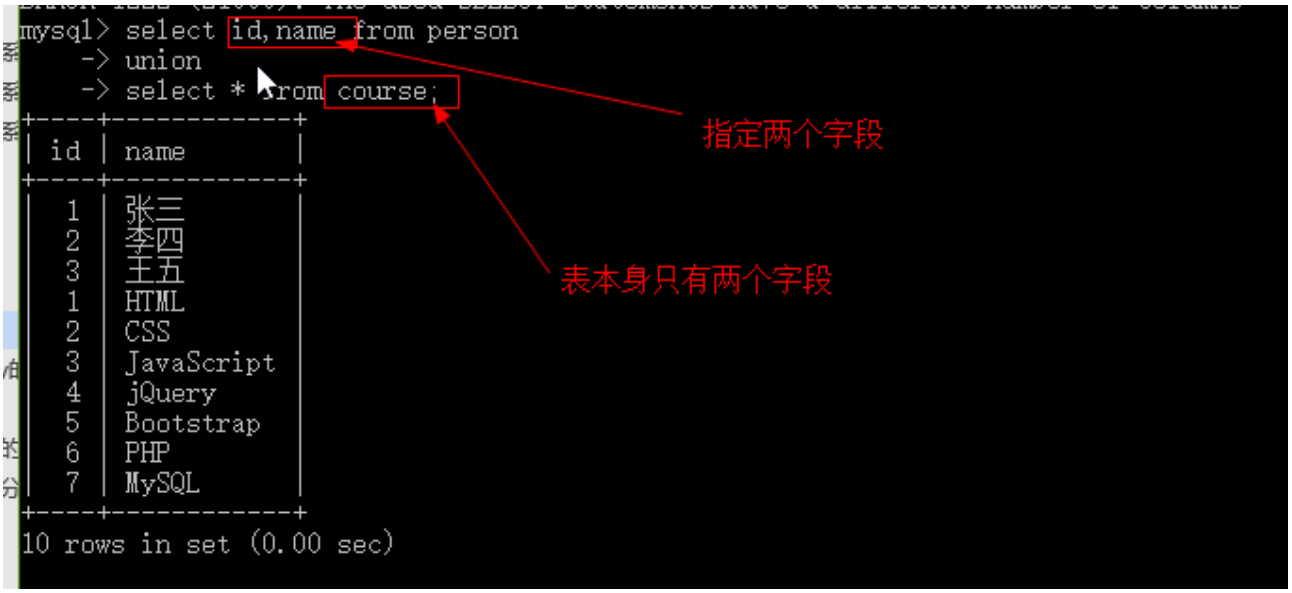
注意细节：

1、联合后，字段显示的以第一个表的字段为准

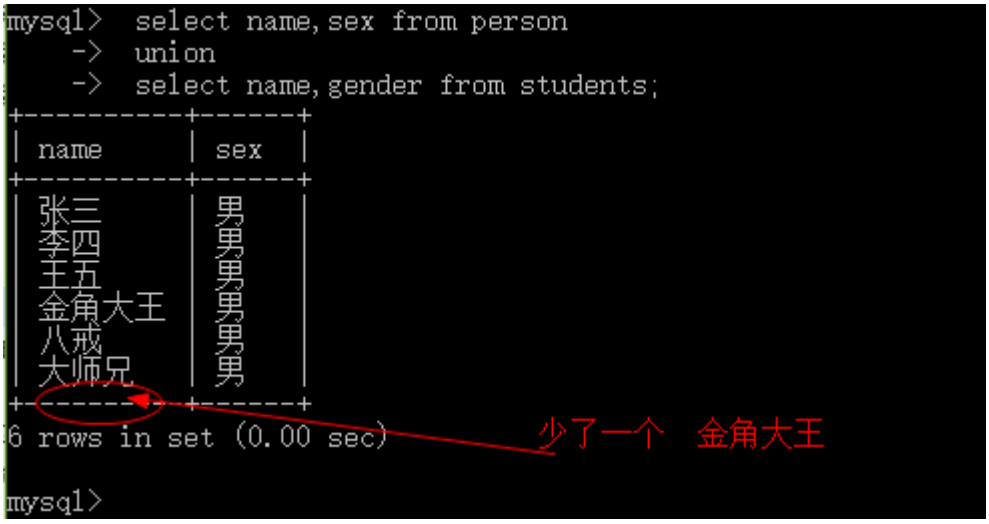
2、如果使用 select * 进行多表联合，必须保证两个表的字段要一致 ， 否则报错：

```
mysql> select * from person
    -> union
    -> select * from course;
ERROR 1222 (21000): The used SELECT statements have a different number of columns
mysql>
```

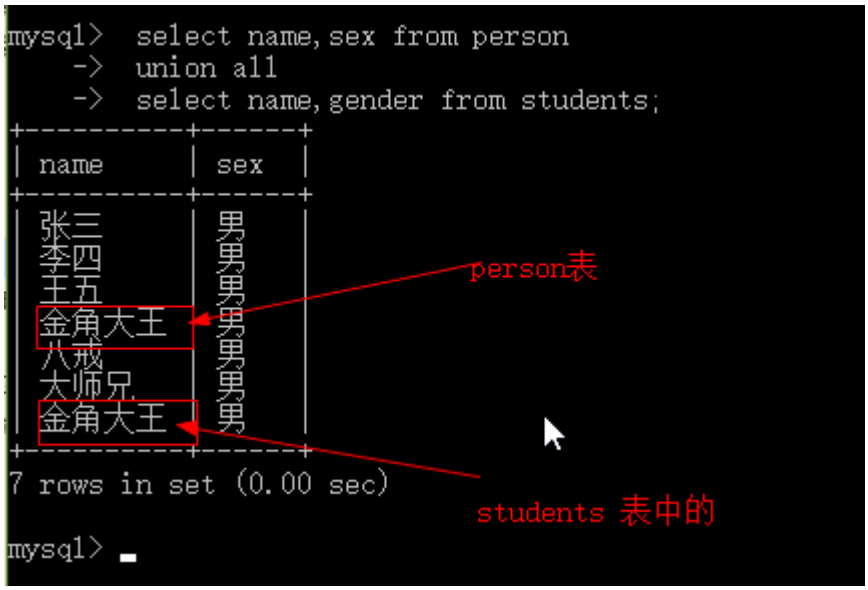
3、如果两个表字段个数不相同，但是又希望联合在一起，此时需要使用 select 指定相同个数的字段



4、union 默认情况下，对查询后联合的结果进行 distinct （去除重复了）



如果不希望，我们可以使用 union all 来进行联合


mysql> select name,sex from person
-> union all
-> select name,gender from students;
name
sex
张三
男
李四
男
王五
男
金角大王
男
八戒
男
大师兄
男
金角大王
男
7 rows in set (0.00 sec)
mysql>
person表
students 表中的

union 理论上只要保证字段数一样，不需要每次拿到的数据对应的字段类型一致。永远只保留第一个 select 语句对应的字段名字。

2.4 Order by 的使用

1、 在联合查询中，如果要使用 order by，那么对应的 select 语句必须使用括号括起来

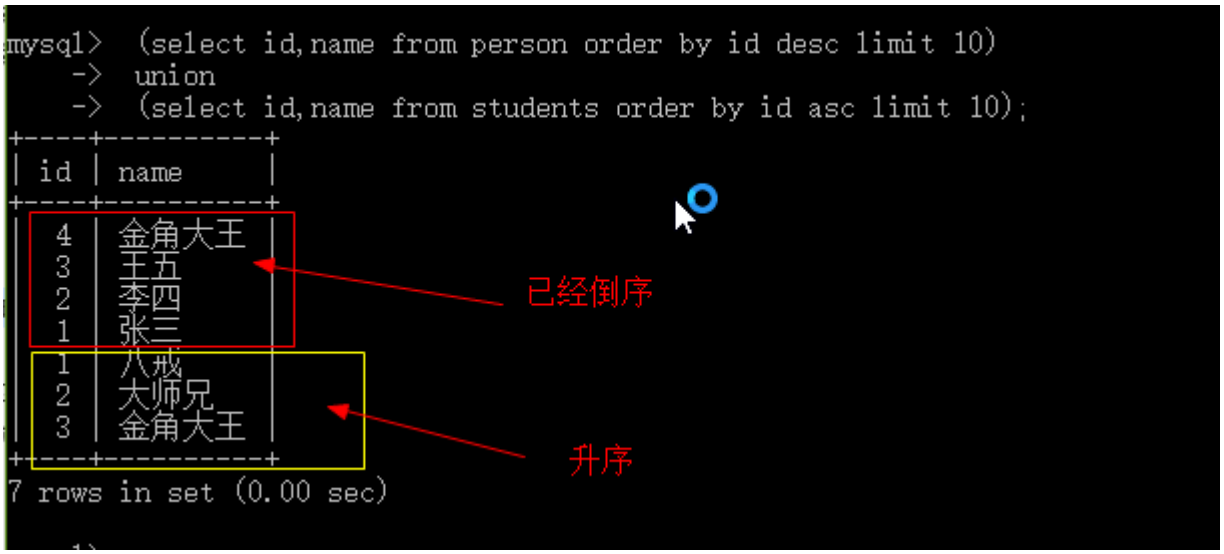
```
    ->  select id,name from students;
+----+----------+
| id | name     |
+----+----------+
|  1 | 张三     |
|  2 | 李四     |
|  3 | 王五     |
|  4 | 金角大王 |
|  1 | 八戒     |
|  2 | 大师兄   |
|  3 | 金角大王 |
+----+----------+
7 rows in set (0.00 sec)
```

```
mysql> select * from member order by id asc
    -> union all
    -> select * from user order by id desc;
ERROR 1221 (HY000): Incorrect usage of UNION and ORDER BY
mysql>
```

正确的语法：加上括号

```
mysql> (select * from member order by id asc)
    -> union all
    -> (select * from user order by id desc);
+----+-------------+----------+
| id | nicheng     | password |
+----+-------------+----------+
|  1 | 老吊线      | 123      |
|  2 | 二师兄说得对 | 123      |
|  1 | 老吊线      | 123      |
|  2 | 孙长老      | 123      |
|  3 | 二师兄      | 123      |
|  4 | 大师兄说得对 | 123      |
|  5 | 农夫三拳    | 123      |
+----+-------------+----------+
7 rows in set (0.00 sec)
```

2、 orderby 在联合查询中若要生效，必须配合使用 limit：而 limit 后面必须跟对应的限制数量（通常可以使用一个较大的值：大于对应表的记录数）



3. 连接查询

连接查询：将多张表连到一起进行查询（会导致记录数行和字段数列发生改变）

3.1 连接查询的意义

在关系型数据库设计过程中，实体（表）与实体之间是存在很多联系的。在关系型数据库表的设计过程中，遵循着关系来设计：一对一，一对多和多对多，通常在实际操作的过程中，需要利用这层关系来保证数据的完整性。

3.2 连接查询分类

连接查询一共有以下几类：
1）交叉连接
2）内连接
3）外连接：左外连接（左连接）和右外连接（右连接）
4）自然连接

4. 交叉连接

交叉连接：将一张表的数据与另外一张表彼此交叉

student

id	name	gender	tel
1	八戒	男	13912345555
2	大师兄	男	13912346666
3	金角大王	男	13922222222

3条

course

id	course_name
1	HTML
2	CSS
3	JavaScript
4	jQuery
5	BootStrap
6	PHP
7	MySQL

student 表和course表进行交叉连接

1）把student 字段和course 字段组成新的结果集字段 student字段数+course表的字段数 7条

2）依次取出student表的数据和course表的每条记录进行连接

八戒	html
大师兄	html
金角大王	html
八戒	css
大师兄	css
金角大王	css
...	...

这种方式也称为：笛卡尔连接
结果：就是笛卡尔积

3）交叉连接后，记录的数目 student(3) * course(7) = 21

4.1 原理

1、 从第一张表依次取出每一条记录
2、 每条记录都会和第二张表的所有记录进行连接
3、 记录数 = 第一张表记录数 * 第二张表记录数；
字段数 = 第一张表字段数 + 第二张表字段数（笛卡尔积）

4.2 语法

4.2.1 基本语法：表 1 cross join 表 2;

把 students 表和 course 表进行连接，写法如下：

```
select * from students cross join course;
```

```
mysql> select * from students cross join course;
+----+----------+--------+-------------+----+-------------+
| id | name     | gender | tel         | id | course_name |
+----+----------+--------+-------------+----+-------------+
|  1 | 八戒     | 男     | 13912345555 |  1 | HTML        |
|  2 | 大师兄   | 男     | 13912346666 |  1 | HTML        |
|  3 | 金角大王 | 男     | 13922222222 |  1 | HTML        |
|  1 | 八戒     | 男     | 13912345555 |  2 | CSS         |
|  2 | 大师兄   | 男     | 13912346666 |  2 | CSS         |
|  3 | 金角大王 | 男     | 13922222222 |  2 | CSS         |
|  1 | 八戒     | 男     | 13912345555 |  3 | JavaScript  |
|  2 | 大师兄   | 男     | 13912346666 |  3 | JavaScript  |
|  3 | 金角大王 | 男     | 13922222222 |  3 | JavaScript  |
|  1 | 八戒     | 男     | 13912345555 |  4 | jQuery      |
|  2 | 大师兄   | 男     | 13912346666 |  4 | jQuery      |
|  3 | 金角大王 | 男     | 13922222222 |  4 | jQuery      |
|  1 | 八戒     | 男     | 13912345555 |  5 | Bootstrap   |
|  2 | 大师兄   | 男     | 13912346666 |  5 | Bootstrap   |
|  3 | 金角大王 | 男     | 13922222222 |  5 | Bootstrap   |
|  1 | 八戒     | 男     | 13912345555 |  6 | PHP         |
|  2 | 大师兄   | 男     | 13912346666 |  6 | PHP         |
|  3 | 金角大王 | 男     | 13922222222 |  6 | PHP         |
|  1 | 八戒     | 男     | 13912345555 |  7 | MySQL       |
|  2 | 大师兄   | 男     | 13912346666 |  7 | MySQL       |
|  3 | 金角大王 | 男     | 13922222222 |  7 | MySQL       |
+----+----------+--------+-------------+----+-------------+
```

4.3 应用

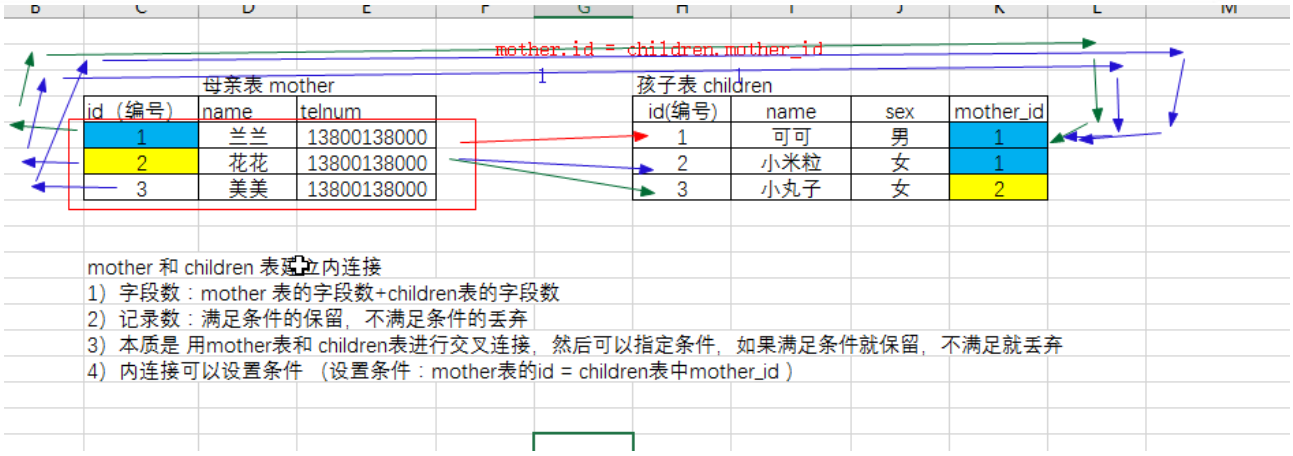
交叉连接产生的结果是笛卡尔积，没有实际应用。
本质：from 表 1,表 2;

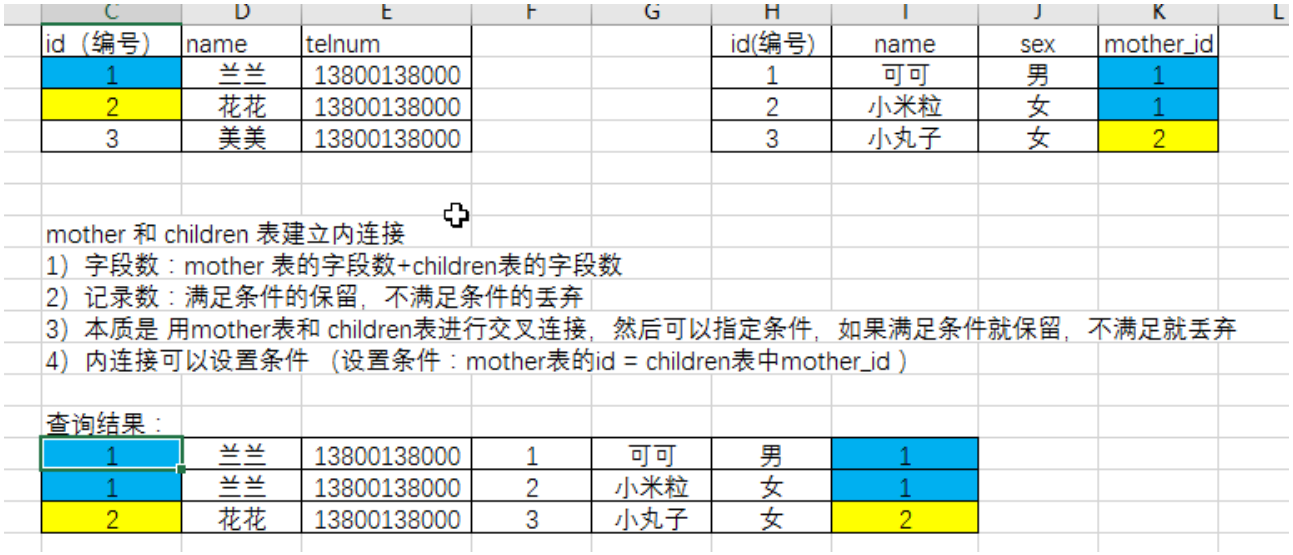
5. 内连接

内连接：inner join，从一张表中取出所有的记录去另外一张表中匹配：利用匹配条件进行匹配，成功了则保留，失败了放弃。

5.1 原理

1、 从第一张表中取出一条记录，然后去另外一张表中进行匹配





2、 利用匹配条件进行匹配：

2.1 匹配到：保留，继续向下匹配

2.2 匹配失败：向下继续，如果全表匹配失败，结束

5.2 语法

基本语法：表 1 [inner] join 表 2 on 匹配条件;

5.2.1 如果内连接没有条件（允许），那么其实就是交叉连接（避免）

```
mysql> use czxy;
Database changed
mysql> select * from mother inner join children;
+----+------+-------------+----+--------+------+-----------+
| id | name | telnum      | id | name   | sex  | mother_id |
+----+------+-------------+----+--------+------+-----------+
|  1 | 兰兰 | 13800138000 |  1 | 可可   | 男   |         1 |
|  2 | 花花 | 13800138000 |  1 | 可可   | 男   |         1 |
|  3 | 美美 | 13800138000 |  1 | 可可   | 男   |         1 |
|  1 | 兰兰 | 13800138000 |  2 | 小米粒 | 女   |         1 |
|  2 | 花花 | 13800138000 |  2 | 小米粒 | 女   |         1 |
|  3 | 美美 | 13800138000 |  2 | 小米粒 | 女   |         1 |
|  1 | 兰兰 | 13800138000 |  3 | 小丸子 | 女   |         2 |
|  2 | 花花 | 13800138000 |  3 | 小丸子 | 女   |         2 |
|  3 | 美美 | 13800138000 |  3 | 小丸子 | 女   |         2 |
+----+------+-------------+----+--------+------+-----------+
9 rows in set (0.00 sec)

mysql>
```

5.2.2 使用匹配条件进行匹配，但可能会报条件“模棱两可”ambiguous-模糊的，不明确的

```
mysql> select * from mother inner join children on id = mother_id;
ERROR 1052 (23000): Column 'id' in on clause is ambiguous
mysql>
```

5.2.3 因为表的设计通常容易产生同名字段，尤其是 ID，所以为了避免重名出现错误，通常使用表名.字段名，来确保唯一性

```
mysql> select * from mother inner join children on mother.id = children.mother_id;
+----+------+-------------+----+--------+------+-----------+
| id | name | telnum      | id | name   | sex  | mother_id |
+----+------+-------------+----+--------+------+-----------+
|  1 | 兰兰 | 13800138000 |  1 | 可可   | 男   |         1 |
|  1 | 兰兰 | 13800138000 |  2 | 小米粒 | 女   |         1 |
|  2 | 花花 | 13800138000 |  3 | 小丸子 | 女   |         2 |
+----+------+-------------+----+--------+------+-----------+
3 rows in set (0.00 sec)
```

5.2.4 通常，如果条件中使用到对应的表名，而表名通常比较长，所以可以通过表别名来简化

```
mysql> select * from mother m inner join children c on m.id = c.mother_id;
+----+------+-------------+----+--------+-----+-----------+
| id | name | telnum      | id | name   | sex | mother_id |
+----+------+-------------+----+--------+-----+-----------+
|  1 | 兰兰 | 13800138000 |  1 | 可可   | 男  |         1 |
|  1 | 兰兰 | 13800138000 |  2 | 小米粒 | 女  |         1 |
|  2 | 花花 | 13800138000 |  3 | 小丸子 | 女  |         2 |
+----+------+-------------+----+--------+-----+-----------+
3 rows in set (0.00 sec)
```

5.2.5 内连接匹配的时候，必须保证匹配到才会保存

比如：此时模拟注册一个用户“农夫三拳”

```
insert into member values(null,'农夫三拳','123');
```

member表

id	nicheng	password
1	老吊线	123
2	二师兄说得对	123456
3	农夫三拳	123
	...	...

日志内容表

id	member_id	type_id	title	content
1	1	1	2月30日养死一条鱼	2月30日养死一条鱼，很伤心
2	1	1	打王者，遇猪队友	坑爹坑爹坑爹啊…
3	1	1	MySQL学的爽	老酸爽了，记不住啊
4	1	1	伟哥最帅	不解释，你们懂得
5	2	1	lol秘籍	带上明月刀砍人必胜！
6	2	1	lol的bug	哈哈，狂刷100万！
7	2	1	班长请吃饭问题	班长你到底请不请
8	2	1	如何让自己长得帅?	把镜子摔了，哈哈
9	2	1	2月31日酸奶一箱	便宜啊，18一箱
10	2	1	班主任太漂亮！	班主任太漂亮，心动了怎么办？

?

因为此时在 blog 表中并没有 “农夫三拳”的匹配，所以内连接后依然不会显示“农夫三拳”

```
mysql> insert into member values(null,'农夫三拳','123');
Query OK, 1 row affected (0.00 sec)

mysql> select * from member as m inner join blog as b on m.id = b.member_id;
+----+--------------+----------+----+-----------+---------+-------------------+--------------------------------------------------------------------+
| id | nicheng      | password | id | member_id | type_id | title             | content                                                            |
+----+--------------+----------+----+-----------+---------+-------------------+--------------------------------------------------------------------+
|  1 | 老吊线       | 123      |  1 |         1 |       1 | 日记：2月30日 晴  | 今天一天都没出太阳，真不好，爸爸买回两条鱼，养在水缸里淹死一条，我很伤心! |
|  1 | 老吊线       | 123      |  2 |         1 |       1 | 打王者，遇猪队友  | 坑爹坑爹坑爹啊…                                                    |
|  1 | 老吊线       | 123      |  3 |         1 |       1 | MySQL学的爽       | 老酸爽了，记不住啊                                                 |
|  1 | 老吊线       | 123      |  4 |         1 |       1 | 伟哥最帅          | 不解释，你们懂得                                                   |
|  2 | 二师兄说得对 | 123      |  5 |         2 |       1 | lol秘籍           | 带上明月刀砍人必胜!                                                |
|  2 | 二师兄说得对 | 123      |  6 |         2 |       1 | lol的bug          | 哈哈，狂刷100万!                                                   |
|  2 | 二师兄说得对 | 123      |  7 |         2 |       1 | 班长请吃饭问题    | 班长你到底请不请                                                   |
|  2 | 二师兄说得对 | 123      |  8 |         2 |       1 | 如何让自己长得帅? | 把镜子摔了，哈哈                                                   |
|  2 | 二师兄说得对 | 123      |  9 |         2 |       1 | 2月31日酸奶一箱   | 便宜啊，18一箱                                                     |
|  2 | 二师兄说得对 | 123      | 10 |         2 |       1 | 班主任太漂亮!     | 班主任太漂亮，心动了怎么办?                                        |
+----+--------------+----------+----+-----------+---------+-------------------+--------------------------------------------------------------------+
```

5.2.6 内连接因为不强制必须使用匹配条件（on）因此可以在数据匹配完成之后，使用 where 条件来限制，效果与 on 一样（建议使用 on）

```
mysql> select * from mother m inner join children c where m.id = c.mother_id and c.name='小米粒';
+----+------+-------------+----+--------+------+-----------+
| id | name | telnum      | id | name   | sex  | mother_id |
+----+------+-------------+----+--------+------+-----------+
|  1 | 兰兰 | 13800138000 |  2 | 小米粒 | 女   |         1 |
+----+------+-------------+----+--------+------+-----------+
1 row in set (0.00 sec)
```

使用 on ... and... 多个条件

5.2.7 内连接更普遍的写法

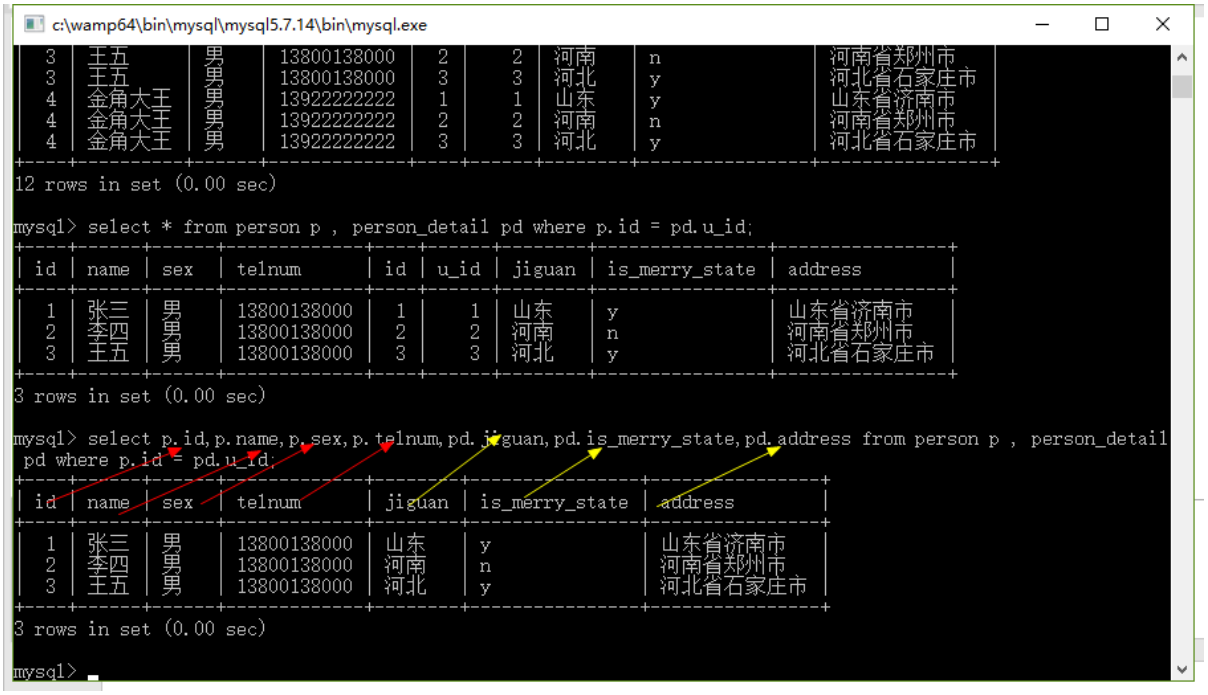
select 字段列表 from 表 a, 表 b where a.字段 = b.字段

```
mysql> select * from mother m ,children c where m.id = c.mother_id;
+----+------+-------------+----+--------+------+-----------+
| id | name | telnum      | id | name   | sex  | mother_id |
+----+------+-------------+----+--------+------+-----------+
|  1 | 兰兰 | 13800138000 |  1 | 可可   | 男   |         1 |
|  1 | 兰兰 | 13800138000 |  2 | 小米粒 | 女   |         1 |
|  2 | 花花 | 13800138000 |  3 | 小丸子 | 女   |         2 |
+----+------+-------------+----+--------+------+-----------+
3 rows in set (0.00 sec)
```

添加条件

```
mysql> select * from mother m ,children c where m.id = c.mother_id and c.name='小米粒';
+----+------+-------------+----+--------+------+-----------+
| id | name | telnum      | id | name   | sex  | mother_id |
+----+------+-------------+----+--------+------+-----------+
|  1 | 兰兰 | 13800138000 |  2 | 小米粒 | 女   |         1 |
+----+------+-------------+----+--------+------+-----------+
1 row in set (0.00 sec)
```

5.2.8 内连接后，可以查询指定字段的信息，而不是所有的信息



5.3 应用

内连接通常是在对数据有精确要求的地方使用：必须保证两种表中都能进行数据匹配。

6. 外连接

外链接：outer join，按照某一张表作为主表（表中所有记录在最后都会保留），根据条件去连接另外一张表，从而得到目标数据。

外连接分为两种：左外连接（left join），右外连接（right join）
左连接：左表是主表
右连接：右表是主表

6.1 原理

母亲表 mother

id（编号）	name	telnum
1	兰兰	13800138000
2	花花	13800138000
3	美美	13800138000

主表

孩子表 children

id(编号)	name	sex	mother_id
1	可可	男	1
2	小米粒	女	1
3	小丸子	女	2

mother 表和 children 表建立外连接
1）外连接，outer join
2）外连接分为：左外连接 left join 和 右外连接 right join
3）外连接有主表，连接的过程是：从主表中取出数据去连接的表中依次查找，如果满足条件则表保留
4）外连接一定会保存主表中的所有的记录
5）连接条件必须写，on mother.id = children.mother_id;
6）关于主表 如果使用的 left join 左边的表就是主表，如果使用的 right join 那么右侧的表是主表
7）如果主表的记录，在连接的表中没有查找到匹配的记录，那么此时主表的记录依然会保存，但是连接的表的信息全为 NULL
8）外连接后，表中的记录等于主表的记录数

6.2 语法

基本语法：
左连接：主表 left join 从表 on 连接条件;
右连接：从表 right join 主表 on 连接条件;

6.2.1 左连接对应的主表数据在左边;右连接对应的主表数据在右边：

```
mysql> select * from mother left join children on mother.id = children.mother_id;
+----+------+-------------+------+--------+------+-----------+
| id | name | telnum      | id   | name   | sex  | mother_id |
+----+------+-------------+------+--------+------+-----------+
|  1 | 兰兰 | 13800138000 |    1 | 可可   | 男   |         1 |
|  1 | 兰兰 | 13800138000 |    2 | 小米粒 | 女   |         1 |
|  2 | 花花 | 13800138000 |    3 | 小丸子 | 女   |         2 |
|  3 | 美美 | 13800138000 | NULL | NULL   | NULL |      NULL |
+----+------+-------------+------+--------+------+-----------+
4 rows in set (0.00 sec)

mysql>
```

6.2.2 右连接查看数据

```
mysql> select * from mother right join children on mother.id = children.mother_id;
+------+------+-------------+----+--------+------+-----------+
| id   | name | telnum      | id | name   | sex  | mother_id |
+------+------+-------------+----+--------+------+-----------+
|    1 | 兰兰 | 13800138000 |  1 | 可可   | 男   |         1 |
|    1 | 兰兰 | 13800138000 |  2 | 小米粒 | 女   |         1 |
|    2 | 花花 | 13800138000 |  3 | 小丸子 | 女   |         2 |
| NULL | NULL |        NULL |  4 | 小爱   | 女   |         4 |
+------+------+-------------+----+--------+------+-----------+
4 rows in set (0.00 sec)
```

外连接的时候，也可以使用别名

```
mysql> select * from mother m right join children c on m.id = c.mother_id;
+------+------+-------------+----+--------+------+-----------+
| id   | name | telnum      | id | name   | sex  | mother_id |
+------+------+-------------+----+--------+------+-----------+
|    1 | 兰兰 | 13800138000 |  1 | 可可   | 男   |         1 |
|    1 | 兰兰 | 13800138000 |  2 | 小米粒 | 女   |         1 |
|    2 | 花花 | 13800138000 |  3 | 小丸子 | 女   |         2 |
| NULL | NULL |        NULL |  4 | 小爱   | 女   |         4 |
+------+------+-------------+----+--------+------+-----------+
4 rows in set (0.00 sec)
```

特点

1、 外连接中主表数据记录一定会保存：连接之后不会出现记录数少于主表（内连接可能）
2、 左连接和右连接其实可以互相转换，但是数据对应的位置（表顺序）会改变

6.3 应用

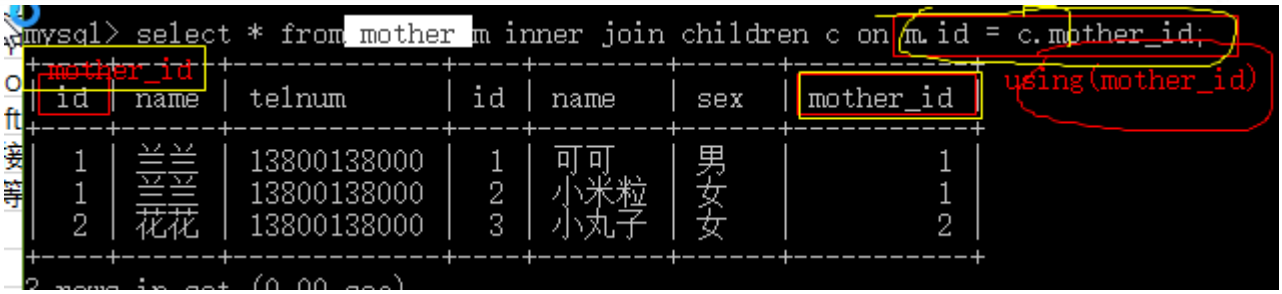
非常常用的一种获取的数据方式：作为数据获取对应主表以及其他数据（关联）

7. Using 关键字

是在连接查询中用来代替对应的 on 关键字的，进行条件匹配。

7.1 原理

1、 在连接查询时，使用 on 的地方用 using 代替
2、 使用 using 的前提是对应的两张表连接的**字段是同名**（类似自然连接自动匹配）



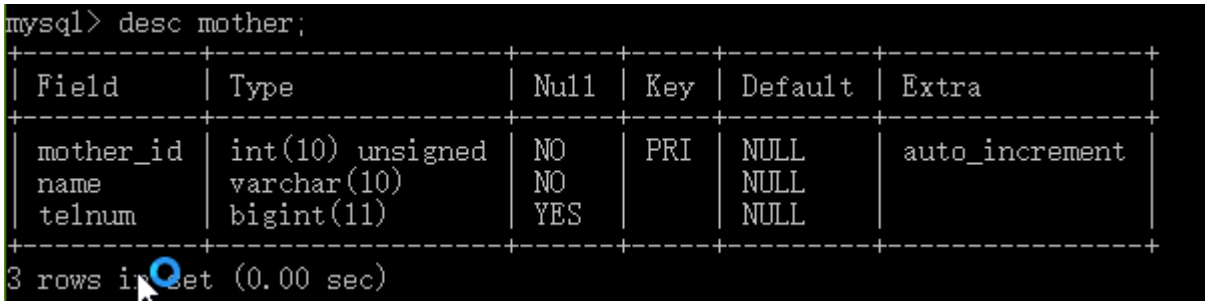
3、 如果使用 using 关键字，那么对应的同名字段，最终在结果中只会保留一个。

7.2 语法

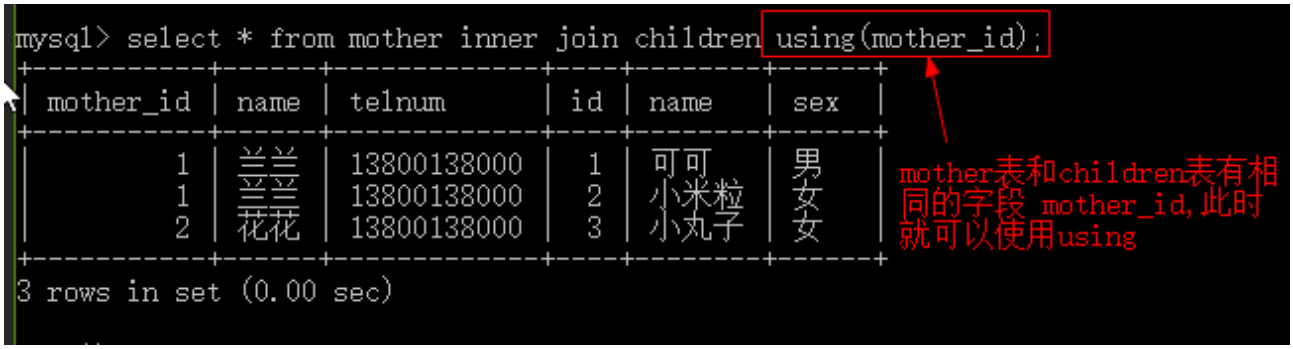
基本语法：表 1 [inner,left,right] join 表 2 using(同名字段列表); //连接字段

第一步：修改 mother 的 id 字段名称为 mother_id(为了和 children 表中的 mother_id 字段一致)

alter table mother change id mother_id int(10) unsigned auto_increment;



7.2.1 使用内连接演示



7.2.2 对比查看结果

```
mysql> select * from mother inner join children using(mother_id);
+-----------+------+-------------+----+--------+------+
| mother_id | name | telnum      | id | name   | sex  |
+-----------+------+-------------+----+--------+------+
|         1 | 兰兰 | 13800138000 |  1 | 可可   | 男   |
|         1 | 兰兰 | 13800138000 |  2 | 小米粒 | 女   |
|         2 | 花花 | 13800138000 |  3 | 小丸子 | 女   |
+-----------+------+-------------+----+--------+------+
3 rows in set (0.00 sec)

mysql> select * from mother inner join children on mother.mother_id = children.mother_id;
+-----------+------+-------------+----+--------+------+-----------+
| mother_id | name | telnum      | id | name   | sex  | mother_id |
+-----------+------+-------------+----+--------+------+-----------+
|         1 | 兰兰 | 13800138000 |  1 | 可可   | 男   |         1 |
|         1 | 兰兰 | 13800138000 |  2 | 小米粒 | 女   |         1 |
|         2 | 花花 | 13800138000 |  3 | 小丸子 | 女   |         2 |
+-----------+------+-------------+----+--------+------+-----------+
3 rows in set (0.00 sec)
```

8. 整库数据备份与还原

整库数据备份也叫 SQL 数据备份：备份的结果都是 SQL 指令

在 Mysql 中提供了一个专门用于备份 SQL 的客户端：mysqldump.exe

8.1 应用场景

SQL 备份是一种 mysql 非常常见的备份与还原方式，SQL 备份不只是备份数据，还备份对应的 SQL 指令（表结构）：即便是数据库遭到毁灭性的破坏（数据库被删），那么利用 SQL 备份依然可以实现数据还原。

SQL 备份因为需要备份结构，因此产生的备份文件特别大，因此不适合特大型数据备份，也不适合数据变换频繁型数据库备份。

8.2 应用方案

8.2.1 SQL 备份

SQL 备份用到的是专门的备份客户端，因此还没与数据库服务器进行连接。

```
基本语法：
mysqldump.exe   -hPup   数据库名字 [表 1   [表 2…]]  >  备份文件地址

-hpup 是指：-h(host) -p(port) -u(user) -p(password)
```

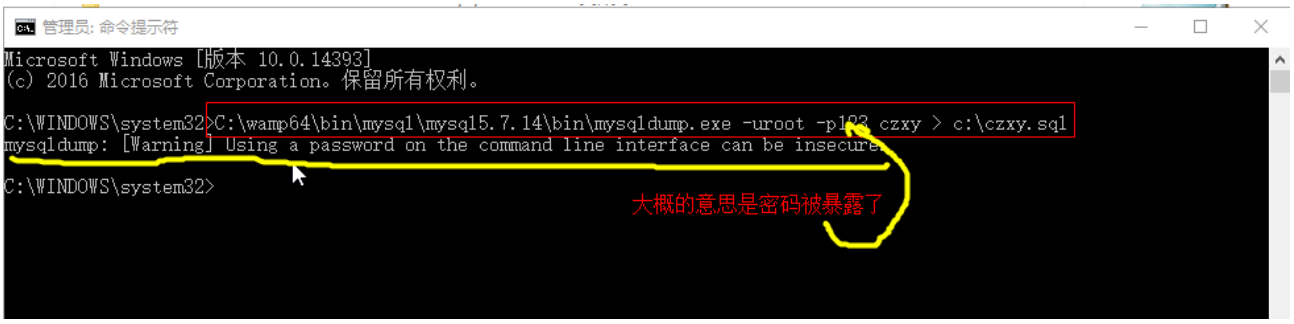
备份可以有三种形式：

- 整库备份（只需要提供数据库名字）

备份数据库需要更高的操作权限，如下图要以管理员身份运行命令行

打开命令提示行，输入导出指令

```
    C:\wamp64\bin\mysql\mysql5.7.14\bin\mysqldump.exe  -uroot  -p123  czxy  >
c:\czxy.sql
```

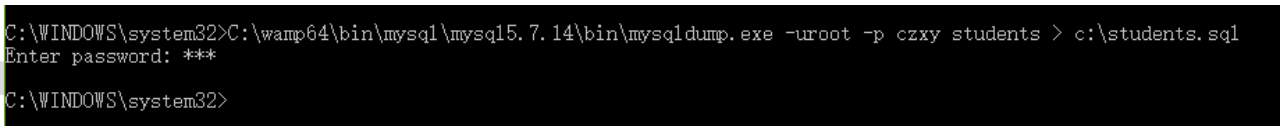


- 单表备份：数据库后面跟一张表

导出时可以不在命令中书写密码，就没有上述的警告了

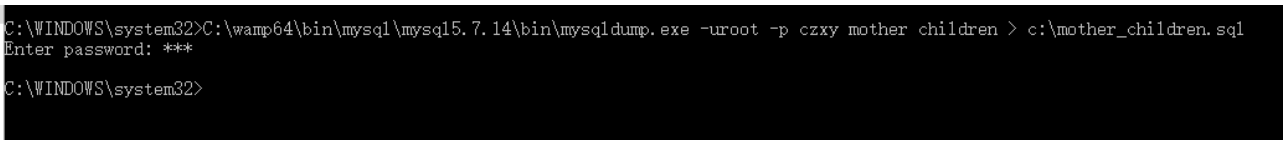
```
    C:\wamp64\bin\mysql\mysql5.7.14\bin\mysqldump.exe -uroot -p czxy students >
c:\students.sql
```

导出单表：

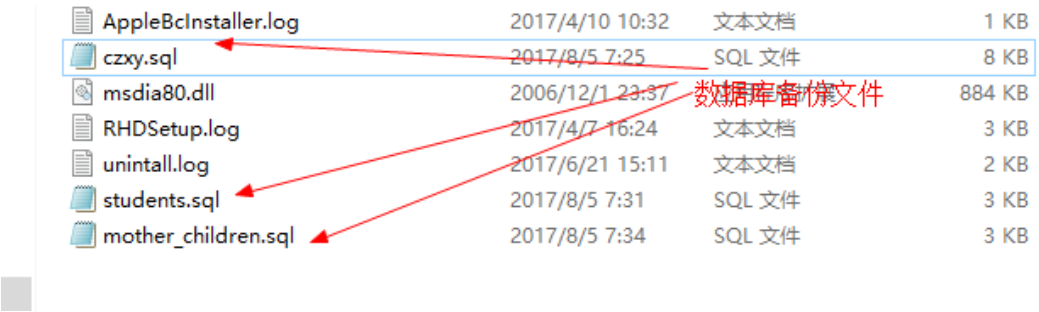


- 多表备份：数据库后跟多张表

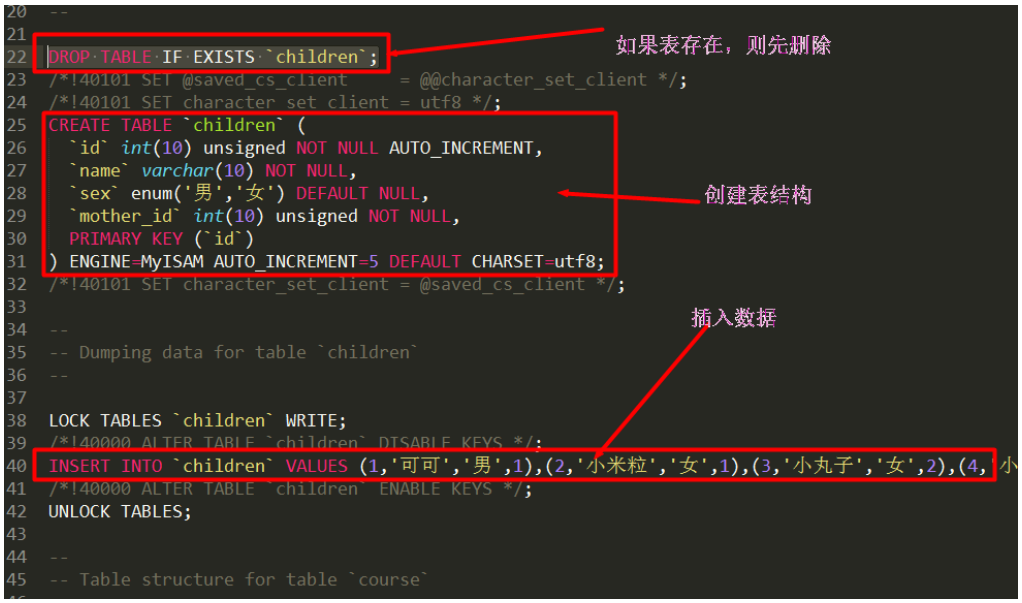
```
    C:\wamp64\bin\mysql\mysql5.7.14\bin\mysqldump.exe -uroot -p czxy mother
children > c:\mother_children.sql
```



查看备份的成果



查看备份文件中的具体内容




```
--
DROP TABLE IF EXISTS `children`;
/*!40101 SET @saved_cs_client     = @@character_set_client */;
/*!40101 SET character_set_client = utf8 */;
CREATE TABLE `children` (
  `id` int(10) unsigned NOT NULL AUTO_INCREMENT,
  `name` varchar(10) NOT NULL,
  `sex` enum('男','女') DEFAULT NULL,
  `mother_id` int(10) unsigned NOT NULL,
  PRIMARY KEY (`id`)
) ENGINE=MyISAM AUTO_INCREMENT=5 DEFAULT CHARSET=utf8;
/*!40101 SET character_set_client = @saved_cs_client */;

--
-- Dumping data for table `children`
--

LOCK TABLES `children` WRITE;
/*!40000 ALTER TABLE `children` DISABLE KEYS */;
INSERT INTO `children` VALUES (1,'可可','男',1),(2,'小米粒','女',1),(3,'小丸子','女',2),(4,'小
/*!40000 ALTER TABLE `children` ENABLE KEYS */;
UNLOCK TABLES;

--
-- Table structure for table `course`
```

8.2.2 数据还原

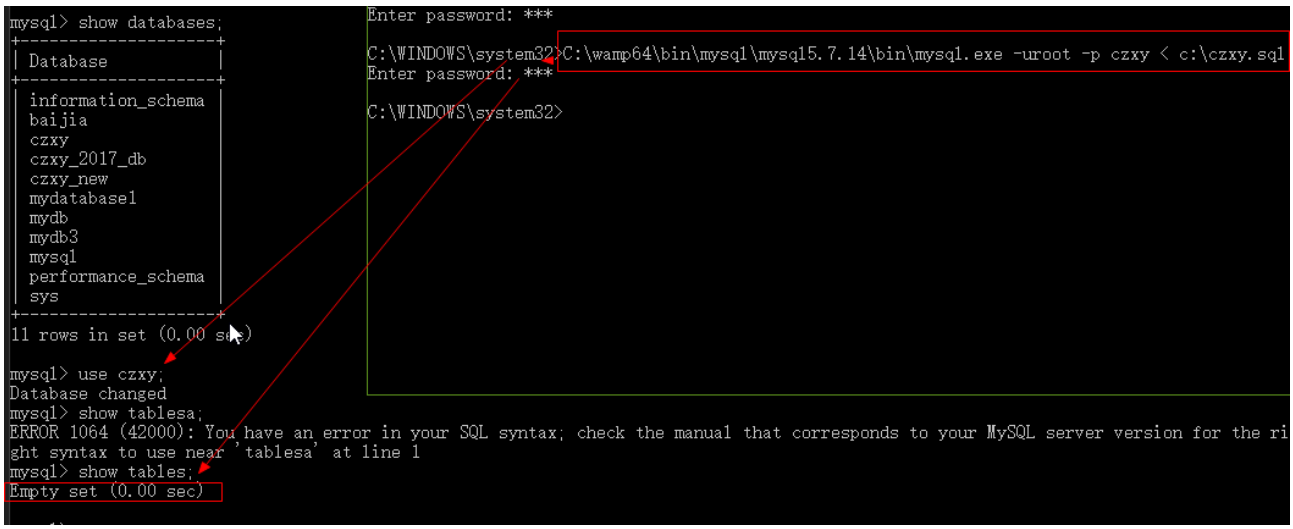
MySQL 提供了多种方式来实现：两种
mysqldump 备份的数据中没有关于数据库本身的操作，都是针对表级别的操作：当进行数据（SQL 还原），必须指定数据库

模拟数据库误删除了

- 利用 mysql.exe 客户端：没有登录之前，可以直接用该客户端进行数据还原

mysql.exe –hPup 数据库 < 文件位置

注意：此时要先创建数据库并且切换到该库中，才能执行还原操作



到入成功！

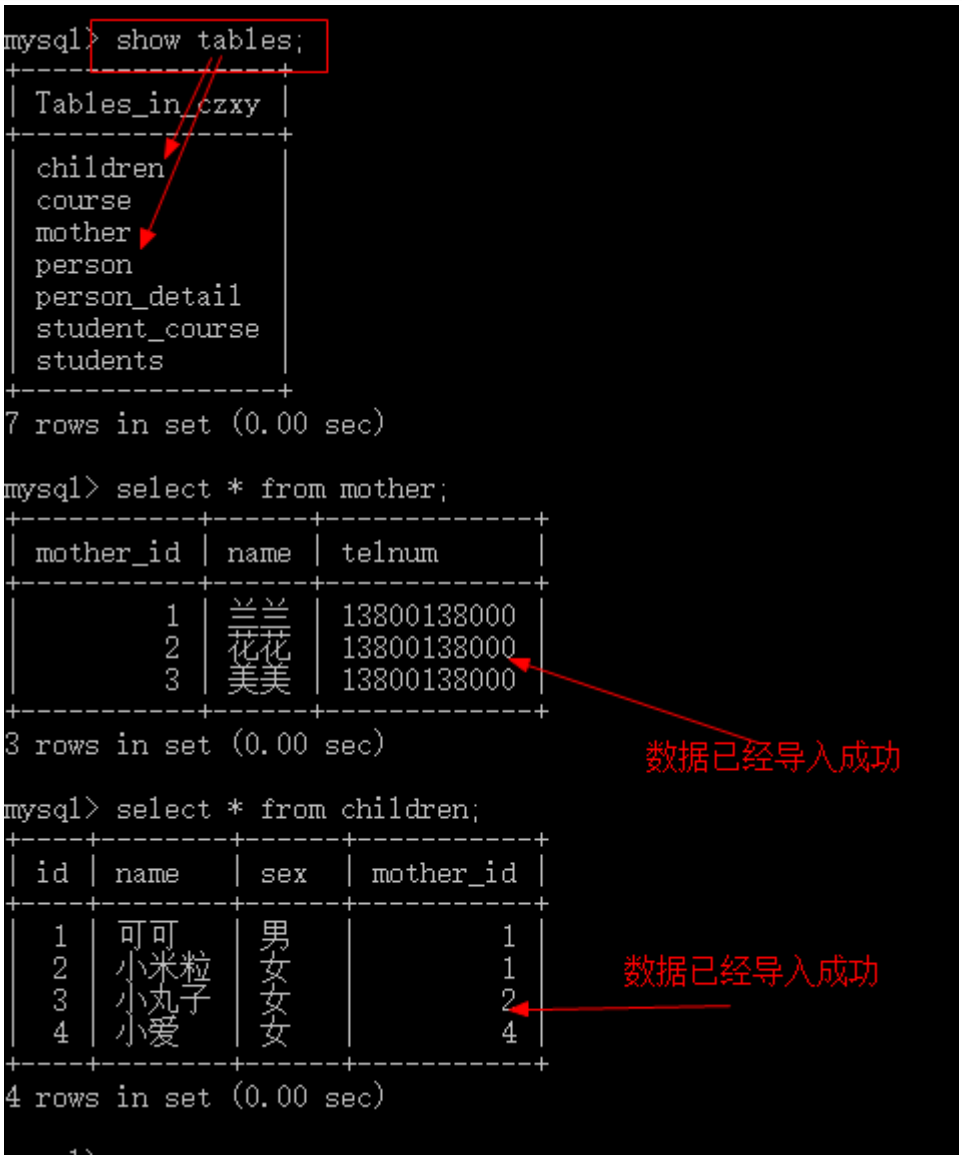
```
mysql> show tables;
+----------------+
| Tables_in_czxy |
+----------------+
| children       |
| course         |
| mother         |
| person         |
| person_detail  |
| student_course |
| students       |
+----------------+
7 rows in set (0.00 sec)

mysql>
```

- 在 SQL 指令，提供了一种导入 SQL 指令的方式

Source　SQL 文件位置;　　//必须先进入到对应的数据库

```
mysql> source c:\mother_children.sql;
ERROR:
Unknown command '\m'.
Query OK, 0 rows affected (0.00 sec)

Query OK, 0 rows affected (0.00 sec)

Query OK, 0 rows affected (0.00 sec)

Query OK, 0 rows affected (0.00 sec)

Query OK, 0 rows affected (0.00 sec)

Query OK, 0 rows affected (0.00 sec)

Query OK, 0 rows affected (0.00 sec)

Query OK, 0 rows affected (0.00 sec)
```

- 人为操作：打开备份文件，复制所有 SQL 指令，然后到 mysql.exe 客户端中去粘贴执行。（不推荐）


mysql> show tables;
+----------------+
| Tables_in_czxy |
+----------------+
| children |
| course |
| mother |
| person |
| person_detail |
| student_course |
| students |
+----------------+
7 rows in set (0.00 sec)
mysql> select * from mother;
+-----------+------+-------------+
| mother_id | name | telnum |
+-----------+------+-------------+
1	兰兰	13800138000
2	花花	13800138000
3	美美	13800138000
+-----------+------+-------------+
3 rows in set (0.00 sec)
数据已经导入成功
mysql> select * from children;
+----+--------+------+-----------+
| id | name | sex | mother_id |
+----+--------+------+-----------+
1	可可	男	1
2	小米粒	女	1
3	小丸子	女	2
4	小爱	女	4
+----+--------+------+-----------+
4 rows in set (0.00 sec)
数据已经导入成功

9. 今日总结