MySQL查询、授权、约束与多表

学习目标

1. 能够使用SQL语句进行条件查询
2. 能够使用SQL语句进行排序
3. 能够使用聚合函数
4. 能够使用SQL语句进行分组查询
5. 能够使用图形界面形式完成数据的备份和恢复
6. 能够使用SQL语句添加主键、外键、唯一、非空约束
7. 能够说出多表之间的关系及其建表原则
8. 能够理解三大范式
9. 能够使用DCL处理MySQL中的用户

第1章 DQL语句

1.1 条件查询

前面我们的查询都是将所有数据都查询出来，但是有时候我们只想获取到`满足条件`的数据 语法格式：`SELECT 字段名 FROM 表名 WHERE 条件;` 流程：取出表中的每条数据，满足条件的记录就返回，不满足条件的记录不返回

1.1.1 准备数据

```
CREATE TABLE student3 (
  id int,
  name varchar(20),
  age int,
  sex varchar(5),
  address varchar(100),
  math int,
  english int
);

INSERT INTO student3(id,NAME,age,sex,address,math,english) VALUES (1,'马云',55,'男','杭州',66,78),(2,'马化腾',45,'女','深圳',98,87),(3,'马景涛',55,'男','香港',56,77),(4,'柳岩',20,'女','湖南',76,65),(5,'柳青',20,'男','湖南',86,NULL),(6,'刘德华',57,'男','香港',99,99),(7,'马德',22,'女','香港',99,99),(8,'德玛西亚',18,'男','南京',56,65);
```

1.1.2 比较运算符

`>`大于 `<`小于 `<=`小于等于 `>=`大于等于 `=`等于 `<>`、`!=`不等于

具体操作：

- 查询math分数大于80分的学生

```
SELECT * FROM student3 WHERE math>80;
```

id	name	age	sex	address	math	english
2	马化腾	45	女	深圳	98	87
6	刘德华	57	男	香港	99	99

- 查询english分数小于或等于80分的学生

```
SELECT * FROM student3 WHERE english<=80;
```

id	name	age	sex	address	math	english
1	马云	51	男	杭州	66	78
3	马景涛	55	男	香港	56	77
4	柳岩	20	女	湖南	76	65

- 查询age等于20岁的学生

```
SELECT * FROM student3 WHERE age=20;
```

id	name	age	sex	address	math	english
4	柳岩	20	女	湖南	76	65
5	柳青	20	男	湖南	(NULL)	(NULL)

- 查询age不等于20岁的学生

```
SELECT * FROM student3 WHERE age!=20;
SELECT * FROM student3 WHERE age<>20;
```

id	name	age	sex	address	math	english
1	马云	51	男	杭州	66	78
2	马化腾	45	女	深圳	98	87
3	马景涛	55	男	香港	56	77
6	刘德华	57	男	香港	99	99

1.1.3 逻辑运算符

`and(&&)` 多个条件同时满足 `or(||)` 多个条件其中一个满足 `not(!)` 不满足

具体操作：

- 查询age大于35且性别为男的学生(两个条件同时满足)

```
SELECT * FROM student3 WHERE  age>35 AND sex='男';
```

id	name	age	sex	address	math	english
1	马云	51	男	杭州	66	78
3	马景涛	55	男	香港	56	77
6	刘德华	57	男	香港	99	99

age大于35且性别为男的学生，两个条件同时满足)

- 查询age大于35或性别为男的学生(两个条件其中一个满足)

```
SELECT * FROM student333 WHERE age>35 OR sex='男';
```

id	name	age	sex	address	math	english
1	马云	51	男	杭州	66	78
2	马化腾	45	女	深圳	98	87
3	马景涛	55	男	香港	56	77
5	柳青	20	男	湖南	(NULL)	(NULL)
6	刘德华	57	男	香港	99	99

age大于35或性别为男的学生，最少其中一个条件满足

- 查询id是1或3或5的学生

```
SELECT * FROM student3 WHERE id=1 OR id=3 OR id=5;
```

id	name	age	sex	address	math	english
1	马云	51	男	杭州	66	78
3	马景涛	55	男	香港	56	77
5	柳青	20	男	湖南	(NULL)	(NULL)

in关键字 语法格式：`SELECT 字段名 FROM 表名 WHERE 字段 in (数据1, 数据2...);` `in` 里面的每个数据都会作为一次条件，只要满足条件的就会显示

具体操作：

- 查询id是1或3或5的学生

```
SELECT * FROM student3 WHERE id IN (1,3,5);
```

id	name	age	sex	address	math	english
1	马云	51	男	杭州	66	78
3	马景涛	55	男	香港	56	77
5	柳青	20	男	湖南	(NULL)	(NULL)

- 查询id不是1或3或5的学生

```
SELECT * FROM student3 WHERE id NOT IN (1,3,5);
```

id	name	age	sex	address	math	english
2	马化腾	45	女	深圳	98	87
4	柳岩	20	女	湖南	76	65
6	刘德华	57	男	香港	99	99

1.1.4 范围

`BETWEEN 值1 AND 值2` 表示从值1到值2范围，包头又包尾 比如：`age BETWEEN 80 AND 100` 相当于：`age>=80 && age<=100`

具体操作：

- 查询english成绩大于等于75，且小于等于90的学生

```
SELECT * FROM student3 WHERE english>=75 AND english<=90;
SELECT * FROM student3 WHERE english BETWEEN 75 AND 90;
```

id	name	age	sex	address	math	english
1	马云	51	男	杭州	66	78
2	马化腾	45	女	深圳	98	87
3	马景涛	55	男	香港	56	77

1.1.5 like

`LIKE`表示模糊查询 `SELECT * FROM 表名 WHERE 字段名 LIKE '通配符字符串';` 满足`通配符字符串`规则的数据就会显示出来 所谓的`通配符字符串`就是`含有通配符的字符串` MySQL通配符有两个： `%`: 表示0个或多个字符(任意个字符) `_`: 表示一个字符

具体操作：

- 查询姓马的学生

```
SELECT * FROM student3 WHERE NAME LIKE '马%';
```

id	name	age	sex	address	math	english
1	马云	51	男	杭州	66	78
2	马化腾	45	女	深圳	98	87
3	马景涛	55	男	香港	56	77

- 查询姓名中包含'德'字的学生

```
SELECT * FROM student3 WHERE NAME LIKE '%德%';
```

id	name	age	sex	address	math	english
6	刘德华	57	男	香港	99	99
7	马德	22	女	(NULL)	(NULL)	(NULL)
8	德玛西亚	18	男	(NULL)	(NULL)	(NULL)

- 查询姓马，且姓名有三个字的学生

```
SELECT * FROM student3 WHERE NAME LIKE '马__';
```

id	name	age	sex	address	math	english
2	马化腾	45	女	深圳	98	87
3	马景涛	55	男	香港	56	77

马开头，后面还有2个字符

1.1.6 排序

通过`ORDER BY`子句，可以将查询出的结果进行排序(排序只是显示方式，不会影响数据库中数据的顺序) `SELECT 字段名 FROM 表名 WHERE 字段=值 ORDER BY 字段名 [ASC|DESC];` ASC: 升序, 默认是升序 DESC: 降序

1.1.6.1 单列排序

单列排序就是使用一个字段排序

具体操作：

- 查询所有数据,使用年龄降序排序

```
SELECT * FROM student3 ORDER BY age DESC;
```

id	name	age	sex	address	math	english
6	刘德华	57	男	香港	99	99
1	马云	55	男	杭州	66	78
3	马景涛	55	男	香港	56	77
2	马化腾	45	女	深圳	98	87
7	马德	22	女	香港	99	99
4	柳岩	20	女	湖南	76	65
5	柳青	20	男	湖南	86	(NULL)
8	德玛西亚	18	男	南京	56	65

1.1.6.2 组合排序

组合排序就是先按第一个字段进行排序，如果第一个字段相同，才按第二个字段进行排序，依次类推。 上面的例子中，年龄是有相同的。当年龄相同再使用math进行排序 `SELECT 字段名 FROM 表名 WHERE 字段=值 ORDER BY 字段名1 [ASC|DESC], 字段名2 [ASC|DESC];`

具体操作：

- 查询所有数据,在年龄降序排序的基础上，如果年龄相同再以数学成绩降序排序

```
SELECT * FROM student3 ORDER BY age DESC, math DESC;
```

id	name	age	sex	address	math	english
6	刘德华	57	男	香港	99	99
1	马云	55	男	杭州	66	78
3	马景涛	55	男	香港	56	77
2	马化腾	45	女	深圳	98	87
7	马德	22	女	香港	99	99
5	柳青	20	男	湖南	86	(NULL)
4	柳岩	20	女	湖南	76	65
8	德玛西亚	18	男	南京	56	65

1.1.7 聚合函数

之前我们做的查询都是横向查询，它们都是根据条件一行一行的进行判断，而使用聚合函数查询是纵向查询，它是对一列的值进行计算，然后返回一个结果值。另外聚合函数会忽略空值

五个聚合函数： `count` ：统计指定列记录数，记录为NULL的不统计 `sum` ：计算指定列的数值和，如果不是数值类型，那么计算结果为0 `max` ：计算指定列的最大值 `min` ：计算指定列的最小值 `avg` ：计算指定列的平均值，如果不是数值类型，那么计算结果为0

聚合函数的使用：写在 SQL语句 `SELECT` 后 `字段名` 的地方 `SELECT 字段名... FROM 表名;` `SELECT COUNT(age) FROM 表名;`

具体操作：

- 查询学生总数

```
SELECT COUNT(english) FROM student3;
```

id	name	age	sex	address	math	english
1	马云	55	男	杭州	66	78
2	马化腾	45	女	深圳	98	87
3	马景涛	55	男	香港	56	77
4	柳岩	20	女	湖南	76	65
5	柳青	20	男	湖南	86	(NULL)
6	刘德华	57	男	香港	99	99
7	马德	22	女	香港	99	99
8	德玛西亚	18	男	南京	56	65

COUNT(english)
7

对于NULL的记录不会统计

我们发现对于NULL的记录不会统计 IFNULL(expr1, expr2)的用法：假如expr1 不为 NULL，则 IFNULL() 的返回值为 expr1; 否则其返回值为expr2

id	name	age	sex	address	math	english
1	马云	55	男	杭州	66	78
2	马化腾	45	女	深圳	98	87
3	马景涛	55	男	香港	56	77
4	柳岩	20	女	湖南	76	65
5	柳青	20	男	湖南	86	(NULL)
6	刘德华	57	男	香港	99	99
7	马德	22	女	香港	99	99
8	德玛西亚	18	男	南京	56	65

直接english查询这个字段的数据

```
SELECT english FROM student3;
```

english
78
87
77
65
(NULL)
99
99
65

查询english字段数据，如果为NULL则使用0

```
SELECT IFNULL(english,0) FROM student3;
```

IFNULL(english,0)
78
87
77
65
0
99
99
65

我们可以利用IFNULL()函数，如果记录为NULL，给个默认值，这样统计的数据就不会遗漏

```
SELECT IFNULL(english,0) FROM student3;
```

COUNT(IFNULL(english,0))
8

```
SELECT COUNT(*) FROM student3;
```

id	name	age	sex	address	math	english
1	马云	55	男	杭州	66	78
2	马化腾	45	女	深圳	98	87
3	马景涛	55	男	香港	56	77
4	柳岩	20	女	湖南	76	65
5	柳青	20	男	湖南	86	(NULL)
6	刘德华	57	男	香港	99	99
7	马德	22	女	香港	99	99
8	德玛西亚	18	男	南京	56	65

COUNT(*)
8

- 查询年龄大于40的总数

```
SELECT COUNT(*) FROM student3 WHERE age>40;
```

id	name	age	sex	address	math	english
1	马云	55	男	杭州	66	78
2	马化腾	45	女	深圳	98	87
3	马景涛	55	男	香港	56	77
4	柳岩	20	女	湖南	76	65
5	柳青	20	男	湖南	86	(NULL)
6	刘德华	57	男	香港	99	99
7	马德	22	女	香港	99	99
8	德玛西亚	18	男	南京	56	65

COUNT(*)
4

- 查询数学成绩总分

```
SELECT SUM(math) FROM student3;
```

id	name	age	sex	address	math	english
1	马云	55	男	杭州	66	78
2	马化腾	45	女	深圳	98	87
3	马景涛	55	男	香港	56	77
4	柳岩	20	女	湖南	76	65
5	柳青	20	男	湖南	86	(NULL)
6	刘德华	57	男	香港	99	99
7	马德	22	女	香港	99	99
8	德玛西亚	18	男	南京	56	65

SUM(math)
636

- 查询数学成绩平均分

```
SELECT AVG(math) FROM student3;
```

id	name	age	sex	address	math	english
1	马云	55	男	杭州	66	78
2	马化腾	45	女	深圳	98	87
3	马景涛	55	男	香港	56	77
4	柳岩	20	女	湖南	76	65
5	柳青	20	男	湖南	86	(NULL)
6	刘德华	57	男	香港	99	99
7	马德	22	女	香港	99	99
8	德玛西亚	18	男	南京	56	65

AVG(math)
79.5000

- 查询数学成绩最高分

```
SELECT MAX(math) FROM student3;
```

id	name	age	sex	address	math	english
1	马云	55	男	杭州	66	78
2	马化腾	45	女	深圳	98	87
3	马景涛	55	男	香港	56	77
4	柳岩	20	女	湖南	76	65
5	柳青	20	男	湖南	86	(NULL)
6	刘德华	57	男	香港	99	99
7	马德	22	女	香港	99	99
8	德玛西亚	18	男	南京	56	65

MAX(math)
99

- 查询数学成绩最低分

```
SELECT MIN(math) FROM student3;
```

id	name	age	sex	address	math	english
1	马云	55	男	杭州	66	78
2	马化腾	45	女	深圳	98	87
3	马景涛	55	男	香港	56	77
4	柳岩	20	女	湖南	76	65
5	柳青	20	男	湖南	86	(NULL)
6	刘德华	57	男	香港	99	99
7	马德	22	女	香港	99	99
8	德玛西亚	18	男	南京	56	65

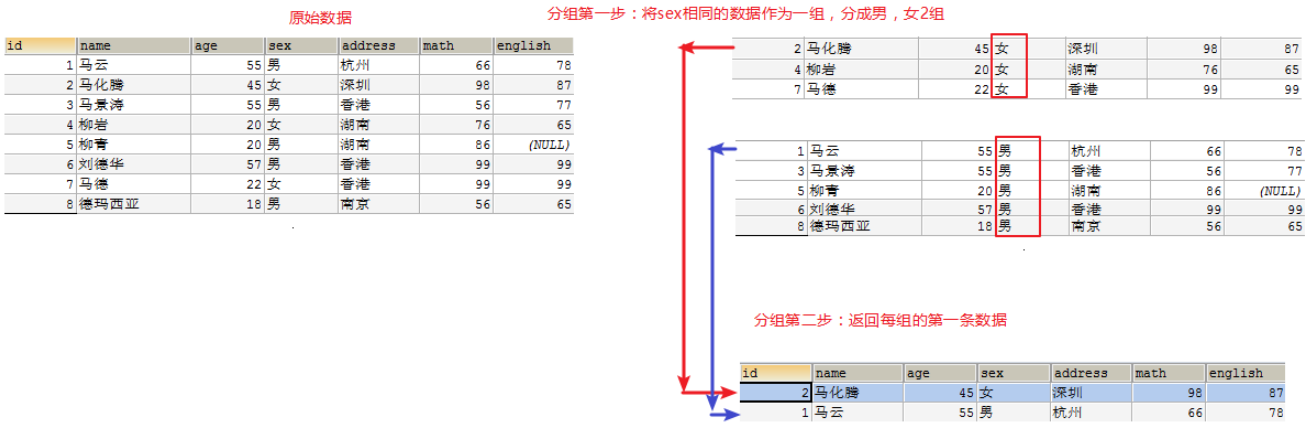
MIN(math)
56

1.1.8 分组

分组查询是指使用 `GROUP BY` 语句对查询信息进行分组，`相同数据作为一组` `SELECT 字段1,字段2... FROM 表名 GROUP BY 分组字段 [HAVING 条件];`

GROUP BY怎么分组的？**将分组字段结果中相同内容作为一组** `SELECT * FROM student3 GROUP BY sex;`

这句话会将sex相同的数据作为一组

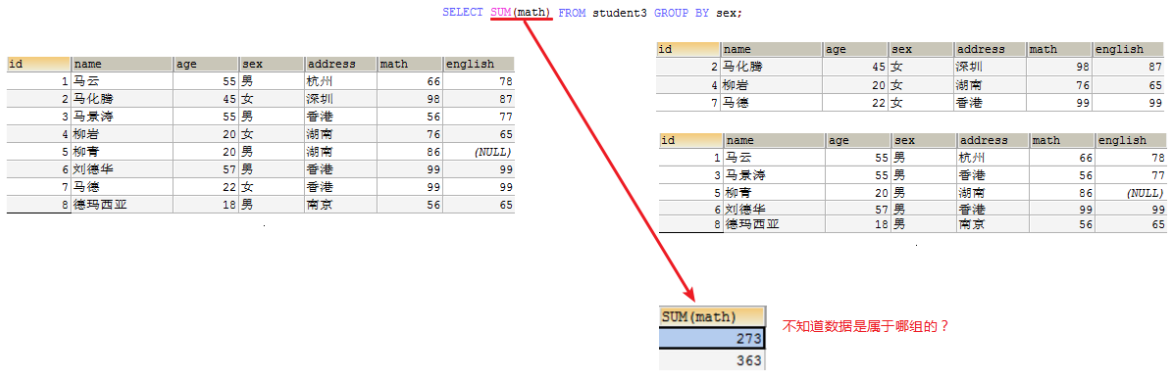
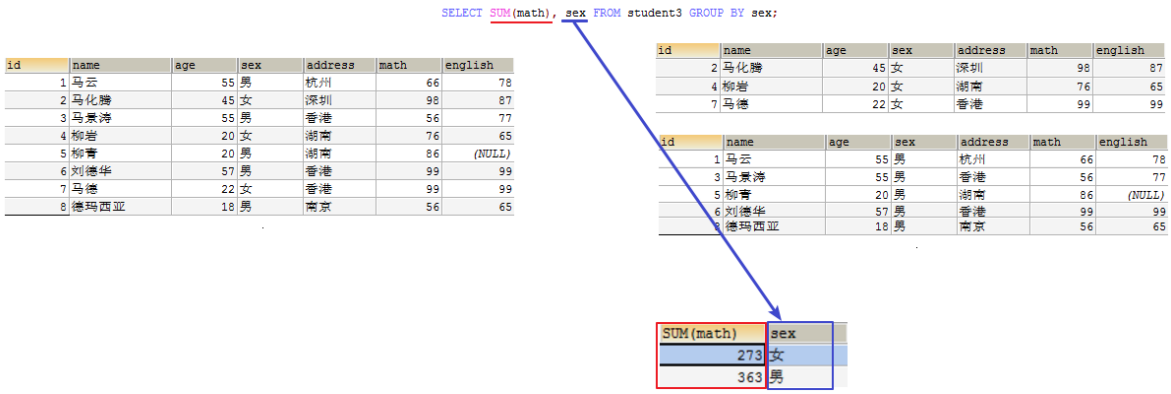


`GROUP BY` 将分组字段结果中相同内容作为一组，并且返回每组的第一条数据，所以单独分组没什么用处。分组的目的就是为了统计，一般分组会跟聚合函数一起使用。

分组后聚合函数的作用？不是操作所有数据，而是操作一组数据。 `SELECT SUM(math), sex FROM student3 GROUP BY sex;` 效果如下：

SUM(math)	sex
273	女
363	男

实际上是将每组的math进行求和,返回每组统计的结果

> 注意事项：当我们使用某个字段分组,在查询的时候也需要将这个字段查询出来,否则看不到数据属于哪组的
>
> - 查询的时候没有查询出分组字段
>
> 
>
>
> - 查询的时候查询出分组字段
>
> 
>

具体步骤：

- 按性别分组

```
SELECT sex FROM student3 GROUP BY sex;
```

id	name	age	sex	address	math	english
1	马云	55	男	杭州	66	78
2	马化腾	45	女	深圳	98	87
3	马景涛	55	男	香港	56	77
4	柳岩	20	女	湖南	76	65
5	柳青	20	男	湖南	86	(NULL)
6	刘德华	57	男	香港	99	99
7	马德	22	女	香港	99	99
8	德玛西亚	18	男	南京	56	65

sex
女
男

相同的数据作为一组

- 查询男女各多少人

```
1.查询所有数据,按性别分组。 2.统计每组人数
SELECT sex, COUNT(*) FROM student3 GROUP BY sex;
```


id	name	age	sex	address	math	english
1	马云	55	男	杭州	66	78
2	马化腾	45	女	深圳	98	87
3	马景涛	55	男	香港	56	77
4	柳岩	20	女	湖南	76	65
5	柳青	20	男	湖南	86	(NULL)
6	刘德华	57	男	香港	99	99
7	马德	22	女	香港	99	99
8	德玛西亚	18	男	南京	56	65

sex
女
男

相同的数据作为一组

- 查询年龄大于25岁的人,按性别分组,统计每组的人数

```
1.先过滤掉年龄小于25岁的人。2.再分组。3.最后统计每组的人数
SELECT sex, COUNT(*) FROM student3 WHERE age > 25 GROUP BY sex;
```

SELECT sex, COUNT(*) FROM student3 WHERE age > 25 GROUP BY sex;

id	name	age	sex	address	math	english
1	马云	55	男	杭州	66	78
2	马化腾	45	女	深圳	98	87
3	马景涛	55	男	香港	56	77
4	柳岩	20	女	湖南	76	65
5	柳青	20	男	湖南	86	(NULL)
6	刘德华	57	男	香港	99	99
7	马德	22	女	香港	99	99
8	德玛西亚	18	男	南京	56	65

1.先过滤掉年龄小于25岁的人。

id	name	age	sex	address	math	english
1	马云	55	男	杭州	66	78
2	马化腾	45	女	深圳	98	87
3	马景涛	55	男	香港	56	77
6	刘德华	57	男	香港	99	99

2.再分组。

id	name	age	sex	address	math	english
2	马化腾	45	女	深圳	98	87

id	name	age	sex	address	math	english
1	马云	55	男	杭州	66	78
3	马景涛	55	男	香港	56	77
6	刘德华	57	男	香港	99	99

3.最后统计每组的人数

sex	COUNT(*)
女	1
男	3

- 查询年龄大于25岁的人,按性别分组,统计每组的人数,并只显示性别人数大于2的数据
  有很多同学可能会将SQL语句写出这样:
  `SELECT sex, COUNT(*) FROM student3 WHERE age > 25 GROUP BY sex WHERE COUNT(*) >2;`

> 注意: 并只显示性别人数>2的数据属于分组后的条件,对于分组后的条件需要使用`having`子句

```
SELECT sex, COUNT(*) FROM student3 WHERE age > 25 GROUP BY sex HAVING COUNT(*) >2;

只有分组后人数大于2的`男`这组数据显示出来
```

查询年龄大于25岁的人,按性别分组,统计每组的人数,并只显示性别人数大于2的数据

```
SELECT sex, COUNT(*) FROM student3 WHERE age > 25 GROUP BY sex HAVING COUNT(*) >2;
```

分组前的条件　　分组后的条件

id	name	age	sex	address	math	english
1	马云	55	男	杭州	66	78
2	马化腾	45	女	深圳	98	87
3	马景涛	55	男	香港	56	77
4	柳岩	20	女	湖南	76	65
5	柳青	20	男	湖南	86	(NULL)
6	刘德华	57	男	香港	99	99
7	马德	22	女	香港	99	99
8	德玛西亚	18	男	南京	56	65

1.先过滤掉年龄小于25岁的人。

id	name	age	sex	address	math	english
1	马云	55	男	杭州	66	78
2	马化腾	45	女	深圳	98	87
3	马景涛	55	男	香港	56	77
6	刘德华	57	男	香港	99	99

2.再分组。

id	name	age	sex	address	math	english
2	马化腾	45	女	深圳	98	87

id	name	age	sex	address	math	english
1	马云	55	男	杭州	66	78
3	马景涛	55	男	香港	56	77
6	刘德华	57	男	香港	99	99

3.统计每组的人数

sex	COUNT(*)
女	1
男	3

4.显示性别人数大于2的数据

sex	COUNT(*)
男	3

> having与where的区别
>
> - having是在分组后对数据进行过滤.
> - where是在分组前对数据进行过滤
> - having后面可以使用聚合函数
> - where后面不可以使用聚合函数

准备数据：

```
INSERT INTO student3(id,NAME,age,sex,address,math,english) VALUES
(9,'唐僧',25,'男','长安',87,78),
(10,'孙悟空',18,'男','花果山',100,66),
(11,'猪八戒',22,'男','高老庄',58,78),
(12,'沙僧',50,'男','流沙河',77,88),
(13,'白骨精',22,'女','白虎岭',66,66),
(14,'蜘蛛精',23,'女','盘丝洞',88,88);
```

1.1.9 limit语句

`LIMIT`是`限制`的意思，所以`LIMIT`的作用就是限制查询记录的条数。 `SELECT *|字段列表 [as 别名] FROM 表名 [WHERE子句] [GROUP BY子句][HAVING子句][ORDER BY子句][LIMIT子句];` 思考：limit子句为什么排在最后？ 因为前面所有的限制条件都处理完了，只剩下显示多少条记录的问题了！

LIMIT语法格式： `LIMIT offset,length;` 或者`limit length;` `offset`是指偏移量，可以认为是跳过的记录数量，默认为0 `length`是指需要显示的总记录数

具体步骤：

- 查询学生表中数据，从第三条开始显示，显示6条

```
我们可以认为跳过前面2条，取6条数据

SELECT * FROM student3 LIMIT 2,6;
```

所有数据

id	name	age	sex	address	math	english
1	马云	55	男	杭州	66	78
2	马化腾	45	女	深圳	98	87
3	马景涛	55	男	香港	56	77
4	柳岩	20	女	湖南	76	65
5	柳青	20	男	湖南	86	(NULL)
6	刘德华	57	男	香港	99	99
7	马德	22	女	香港	99	99
8	德玛西亚	18	男	南京	56	65
9	唐僧	25	男	长安	87	78
10	孙悟空	18	男	花果山	100	66
11	猪八戒	22	男	高老庄	58	78
12	沙僧	50	男	流沙河	77	88
13	白骨精	22	女	白虎岭	66	66
14	蜘蛛精	23	女	盘丝洞	88	88

跳过2条记录　显示6条记录

SELECT * FROM student3 LIMIT 2,6;

id	name	age	sex	address	math	english
3	马景涛	55	男	香港	56	77
4	柳岩	20	女	湖南	76	65
5	柳青	20	男	湖南	86	(NULL)
6	刘德华	57	男	香港	99	99
7	马德	22	女	香港	99	99
8	德玛西亚	18	男	南京	56	65

LIMIT的使用场景：分页 比如我们登录京东，淘宝，返回的商品信息可能有几万条，不是一次全部显示出来。是一页显示固定的条数。 假设我们一每页显示5条记录的方式来分页，SQL语句如下：

```
-- 每页显示5条
-- 第一页： LIMIT 0,5;  跳过0条，显示5条
-- 第二页： LIMIT 5,5;  跳过5条，显示5条
-- 第三页： LIMIT 10,5; 跳过10条，显示5条
SELECT * FROM student3 LIMIT 0,5;
SELECT * FROM student3 LIMIT 5,5;
SELECT * FROM student3 LIMIT 10,5;
```

所有数据

id	name	age	sex	address	math	english
1	马云	55	男	杭州	66	78
2	马化腾	45	女	深圳	98	87
3	马景涛	55	男	香港	56	77
4	柳岩	20	女	湖南	76	65
5	柳青	20	男	湖南	86	(NULL)
6	刘德华	57	男	香港	99	99
7	马德	22	女	香港	99	99
8	德玛西亚	18	男	南京	56	65
9	唐僧	25	男	长安	87	78
10	孙悟空	18	男	花果山	100	66
11	猪八戒	22	男	高老庄	58	78
12	沙僧	50	男	流沙河	77	88
13	白骨精	22	女	白虎岭	66	66
14	蜘蛛精	23	女	盘丝洞	88	88

注意：LIMIT 10，5; -- 不够5条，有多少显示多少

SELECT * FROM student3 LIMIT 0,5;

id	name	age	sex	address	math	english
1	马云	55	男	杭州	66	78
2	马化腾	45	女	深圳	98	87
3	马景涛	55	男	香港	56	77
4	柳岩	20	女	湖南	76	65
5	柳青	20	男	湖南	86	(NULL)

SELECT * FROM student3 LIMIT 5,5;

id	name	age	sex	address	math	english
6	刘德华	57	男	香港	99	99
7	马德	22	女	香港	99	99
8	德玛西亚	18	男	南京	56	65
9	唐僧	25	男	长安	87	78
10	孙悟空	18	男	花果山	100	66

SELECT * FROM student3 LIMIT 10,5;

id	name	age	sex	address	math	english
11	猪八戒	22	男	高老庄	58	78
12	沙僧	50	男	流沙河	77	88
13	白骨精	22	女	白虎岭	66	66
14	蜘蛛精	23	女	盘丝洞	88	88

> **注意**：
>
> - 如果第一个参数是0可以简写：
> `SELECT * FROM student3 LIMIT 0,5;`
> `SELECT * FROM student3 LIMIT 5;`
> - LIMIT 10，5; -- 不够5条，有多少显示多少

第2章 数据库备份

2.1 备份的应用场景

在服务器进行数据传输、数据存储和数据交换，就有可能产生数据故障。比如发生意外停机或存储介质损坏。这时，如果没有采取数据备份和数据恢复手段与措施，就会导致数据的丢失，造成的损失是无法弥补与估量的。

2.2 source命令备份与还原

备份格式：`mysqldump -u用户名 -p密码 数据库 > 文件的路径`

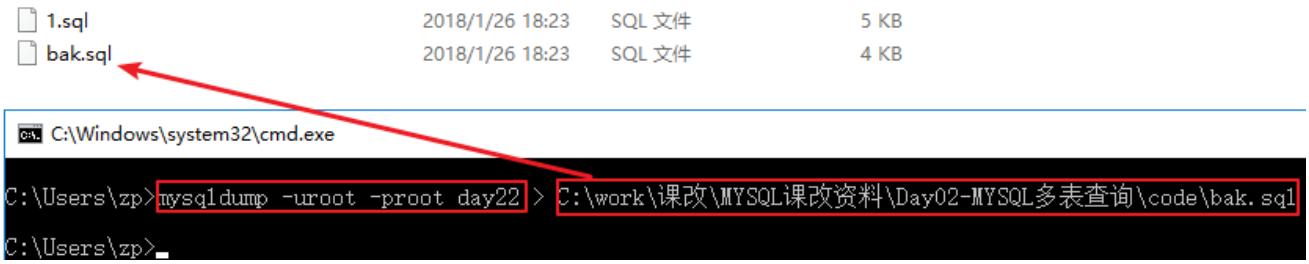
还原格式：`SOURCE 导入文件的路径;`

> 注意：还原的时候需要先登录MySQL,并选中对应的数据库

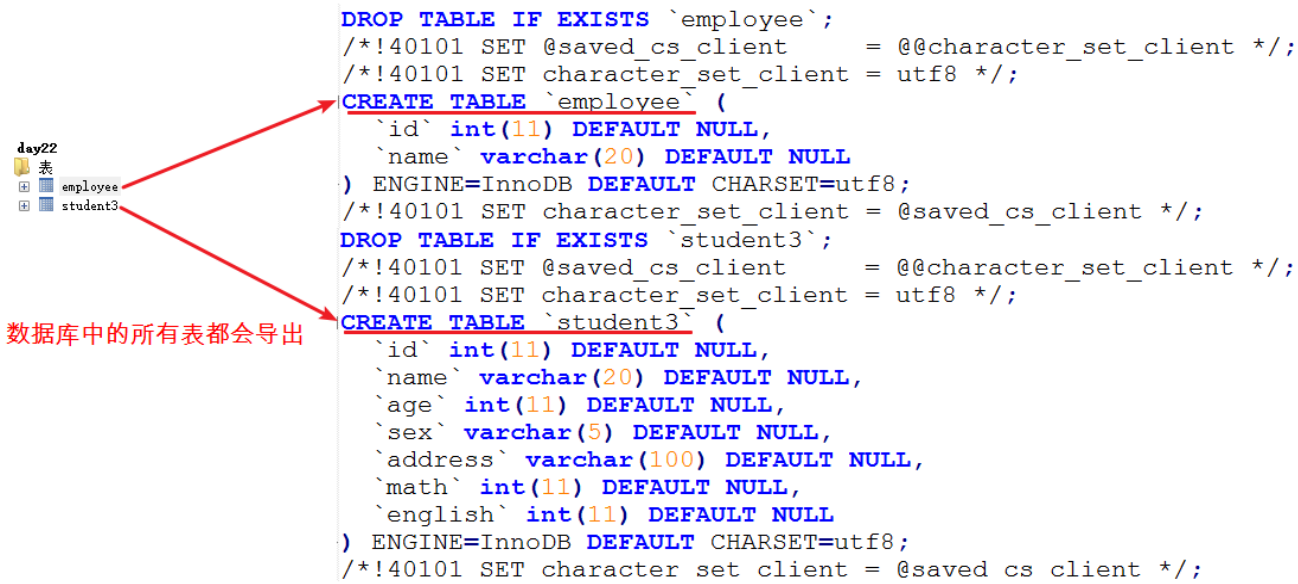
具体操作：

- 备份day22数据库中的数据

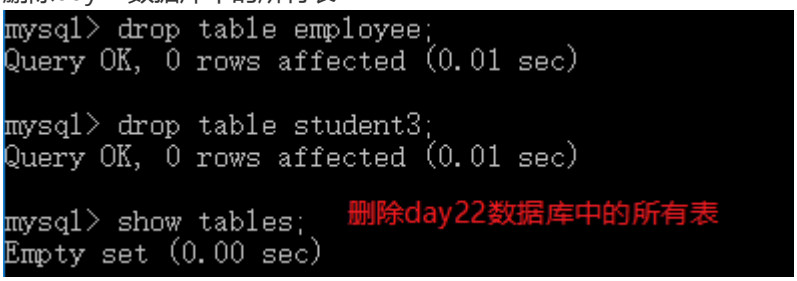
```
mysqldump -uroot -proot day22 > C:\work\课改\MYSQL课改资料\Day02-MYSQL多表查询\code\bak.sql
```



数据库中的所有表和数据都会导出成SQL语句



- 还原day22数据库中的数据
  - 删除day22数据库中的所有表

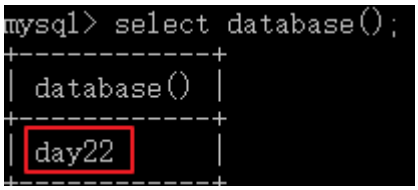


  - 登录MySQL

```
mysql -uroot -proot
```

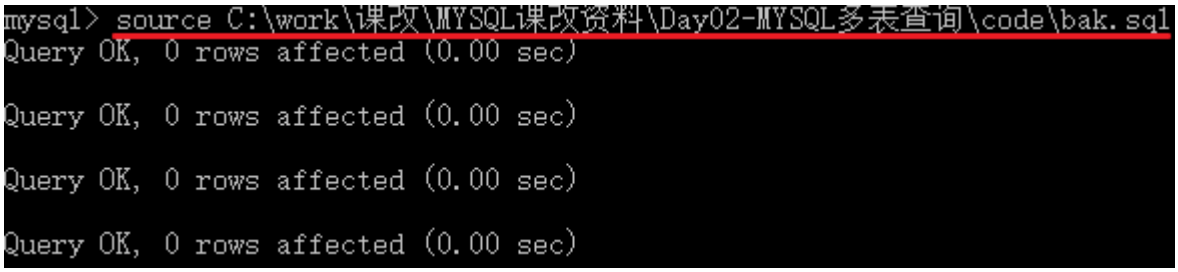
- 选中数据库

```
use day22;
select database();
```



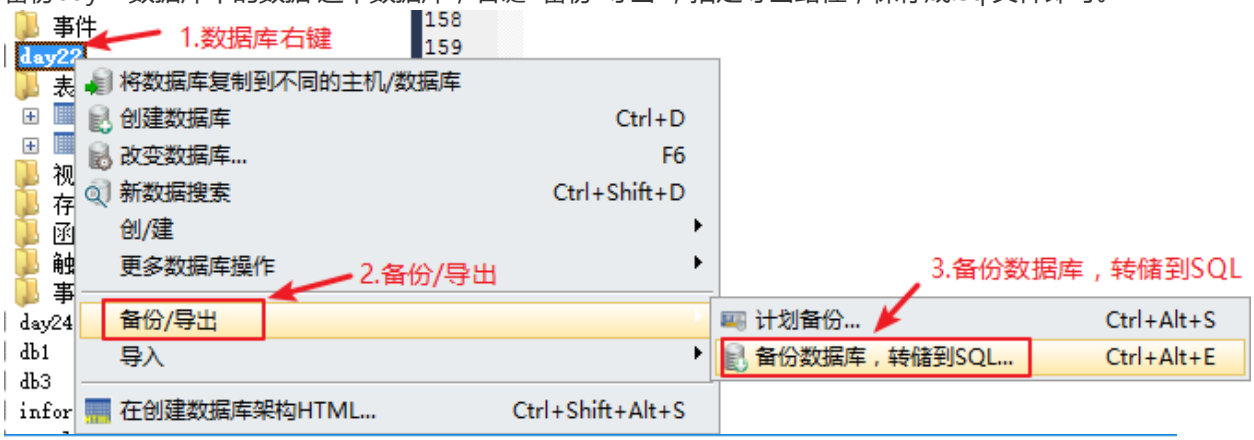
- 使用SOURCE命令还原数据

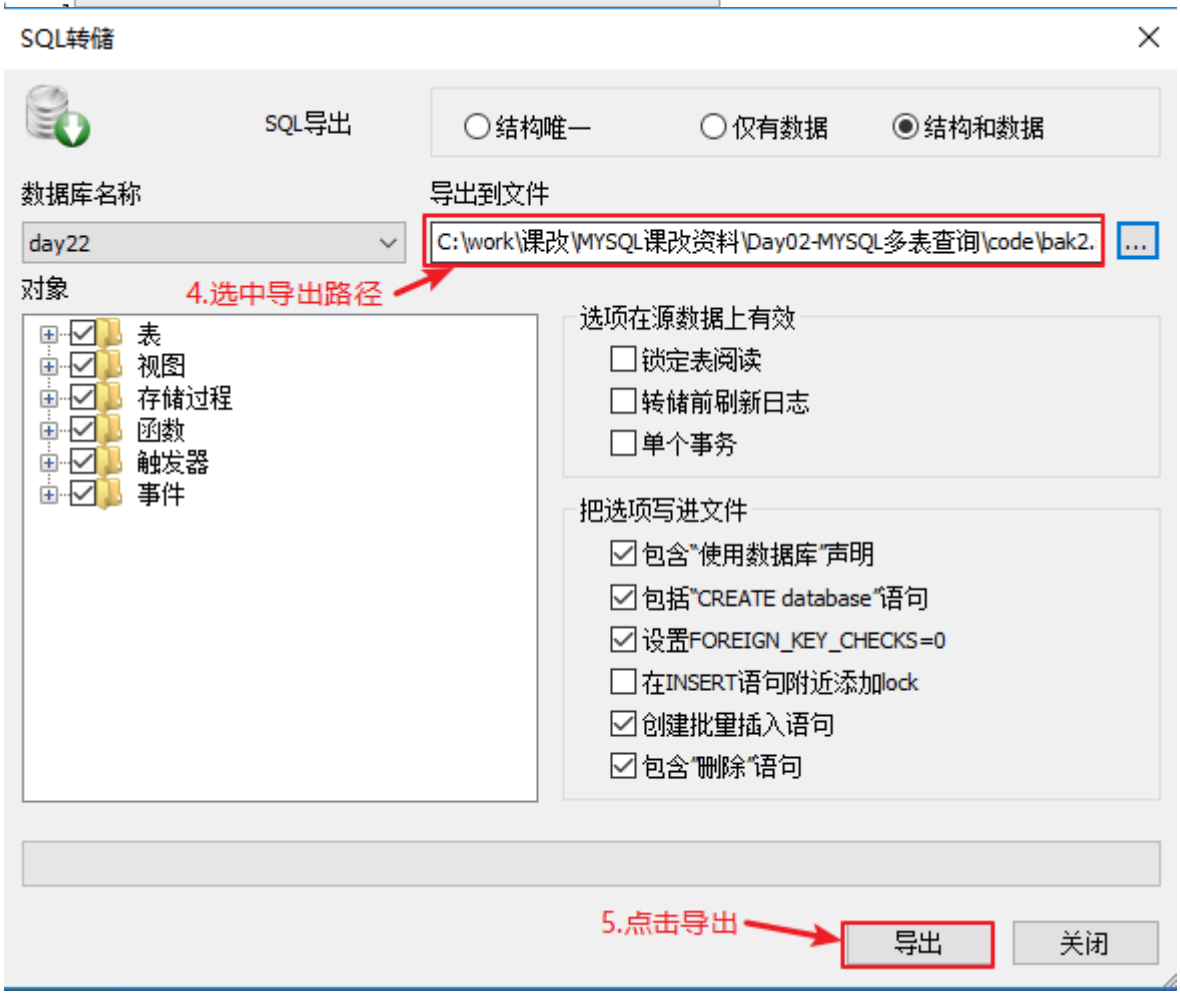
```
source C:\work\课改\MYSQL课改资料\Day02-MYSQL多表查询\code\bak.sql
```



2.3 图形化界面备份与还原

- 备份day22数据库中的数据 选中数据库，右键 "备份/导出"，指定导出路径，保存成.sql文件即可。



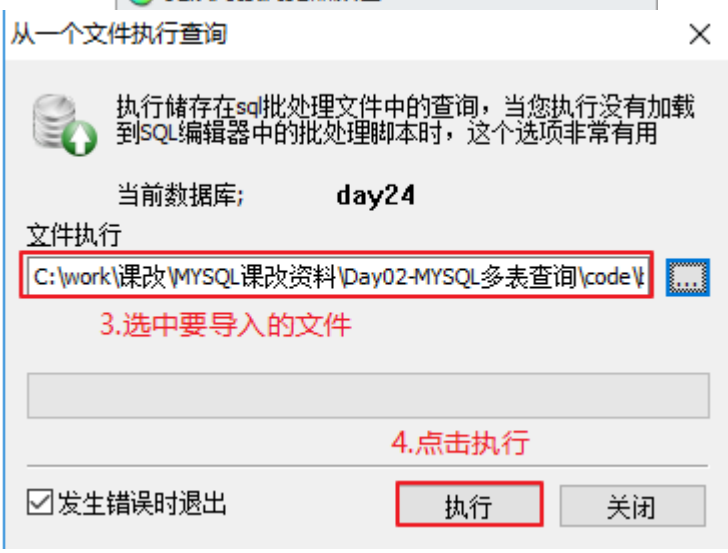


包含创建数据库的语句

```
bak2.sql
/*!40101 SET @OLD_SQL_MODE=@@SQL_MODE, SQL_MODE='NO_AUTO_VALUE_ON_ZERO' */;
/*!40111 SET @OLD_SQL_NOTES=@@SQL_NOTES, SQL_NOTES=0 */;
CREATE DATABASE /*!32312 IF NOT EXISTS*/`day22` /*!40100 DEFAULT CHARACTER SET utf8 */;
包含创建数据库的语句
USE `day22`;

/*Table structure for table `employee` */

DROP TABLE IF EXISTS `employee`;

CREATE TABLE `employee` (
  `id` int(11) DEFAULT NULL,
  `name` varchar(20) DEFAULT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8;
```

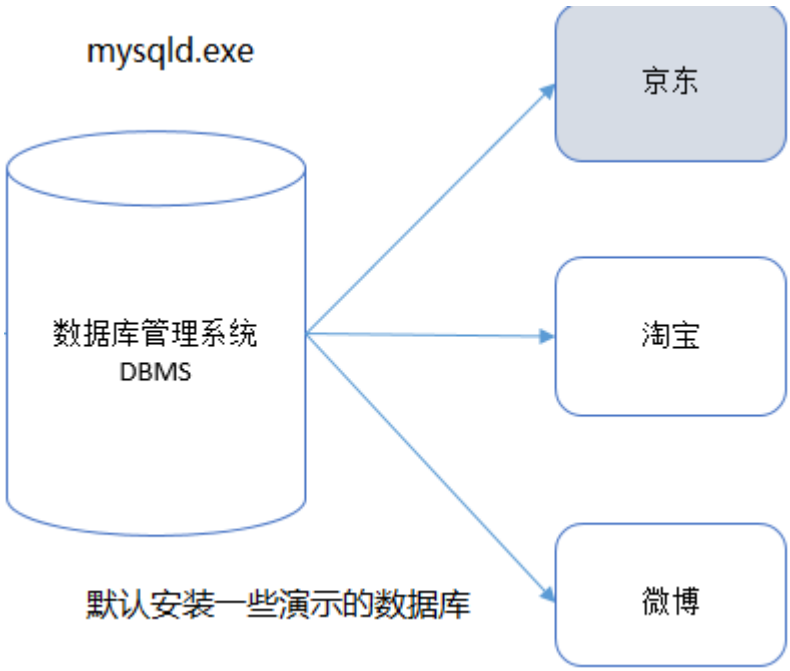
- 还原day22数据库中的数据

- 删除day22数据库
- 数据库列表区域右键“执行SQL脚本”，指定要执行的SQL文件，执行即可



第3章 DCL

我们现在默认使用的都是root用户，超级管理员，拥有全部的权限。但是，一个公司里面的数据库服务器上面可能同时运行着很多个项目的数据库。所以，我们应该可以根据不同的项目建立不同的用户，分配不同的权限来管理和维护数据库。



3.1 创建用户

`CREATE USER '用户名'@'主机名' IDENTIFIED BY '密码';` **关键字说明：**

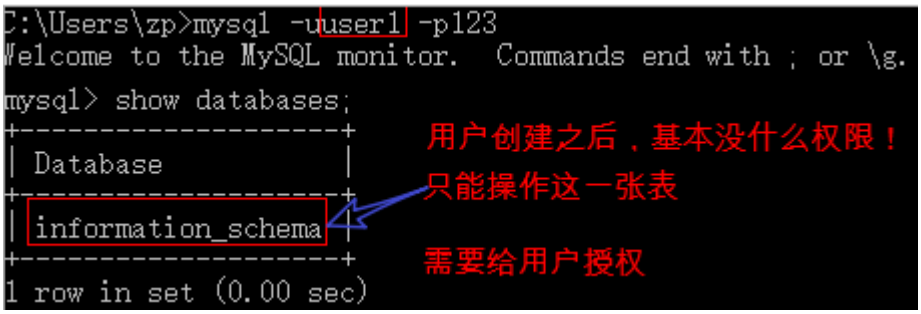
```
1. `用户名`：将创建的用户名
2. `主机名`：指定该用户在哪个主机上可以登陆，如果是本地用户可用localhost，如果想让该用户可以从任意远程主机登陆，可以使用通配符%
3. `密码`：该用户的登陆密码，密码可以为空，如果为空则该用户可以不需要密码登陆服务器
```

具体操作：

```
-- user1用户只能在localhost这个IP登录mysql服务器
CREATE USER 'user1'@'localhost' IDENTIFIED BY '123';
-- user2用户可以在任何电脑上登录mysql服务器
CREATE USER 'user2'@'%' IDENTIFIED BY '123';
```

3.2 授权用户

用户创建之后，基本没什么权限！需要给用户授权



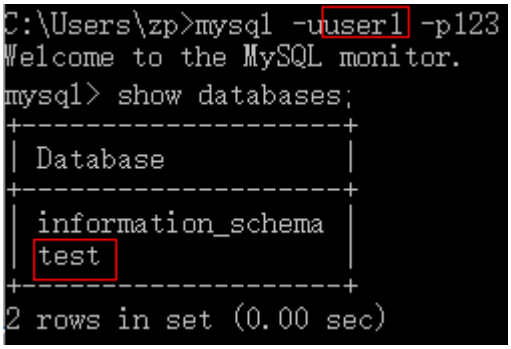
授权格式： `GRANT 权限1, 权限2... ON 数据库名.表名 TO '用户名'@'主机名';` **关键字说明：**

```
1. `GRANT` 授权关键字
2. 授予用户的权限，如`SELECT`，`INSERT`，`UPDATE`等。如果要授予所的权限则使用`ALL`
3. `数据库名.表名`：该用户可以操作哪个数据库的哪些表。如果要授予该用户对所有数据库和表的相应操作权限则可用*表示，如`*.*`
4. `'用户名'@'主机名'`: 给哪个用户授权
```

具体操作：

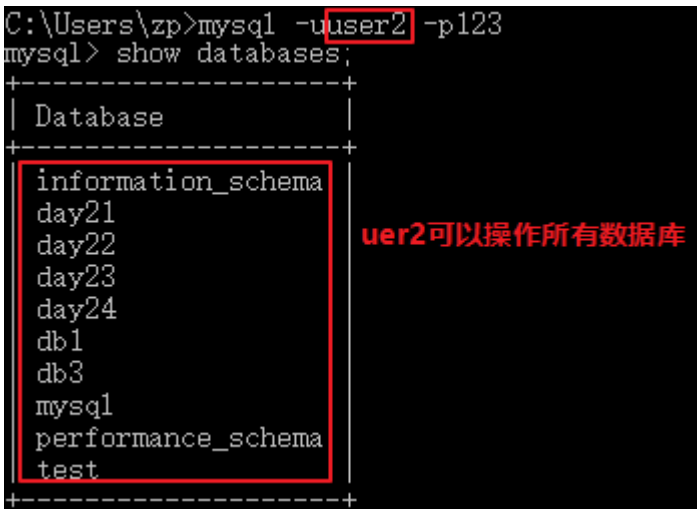
1. 给user1用户分配对test这个数据库操作的权限

```
GRANT CREATE,ALTER,DROP,INSERT,UPDATE,DELETE,SELECT ON test.* TO 'user1'@'localhost';
```



2. 给user2用户分配对所有数据库操作的权限

```
GRANT ALL ON *.* TO 'user2'@'%';
```



3.3 撤销授权

`REVOKE 权限1, 权限2... ON 数据库.表名 FROM '用户名'@'主机名';`

具体操作：

- 撤销user1用户对test操作的权限

```
REVOKE ALL ON test.* FROM 'user1'@'localhost';
```

```
mysql> show databases;
+--------------------+
| Database           |
+--------------------+
| information_schema |
| test               |
+--------------------+
2 rows in set (0.00 sec)
mysql> REVOKE CREATE,ALTER,DROP ON test.* FROM 'user1'@'localhost';
Query OK, 0 rows affected (0.00 sec)
mysql> show databases;
+--------------------+
| Database           |
+--------------------+
| information_schema |
+--------------------+
1 row in set (0.00 sec)
```

取消授权

不能操作test数据库了

3.4 查看权限

`SHOW GRANTS FOR '用户名'@'主机名';` **具体操作：**

- 查看user1用户的权限

```
SHOW GRANTS FOR 'user1'@'localhost';
```

```
mysql> SHOW GRANTS FOR 'user1'@'localhost';
+-----------------------------------------------------------------------------------------------------------------------+
| Grants for user1@localhost                                                                                            |
+-----------------------------------------------------------------------------------------------------------------------+
| GRANT USAGE ON *.* TO 'user1'@'localhost' IDENTIFIED BY PASSWORD '*23AE809DDACAF96AF0FD78ED04B6A265E05AA257'          |
| GRANT SELECT, INSERT, UPDATE, DELETE, CREATE, DROP, ALTER ON `test`.* TO 'user1'@'localhost'                          |
+-----------------------------------------------------------------------------------------------------------------------+
2 rows in set (0.00 sec)
```

user1的权限

3.5 删除用户

`DROP USER '用户名'@'主机名';` **具体操作：**

- 删除user2

```
DROP USER 'user2'@'%';
```

```
mysql> DROP USER 'user2'@'%';
Query OK, 0 rows affected (0.00 sec)
C:\Users\zp>mysql -uuser2 -p123
ERROR 1045 (28000): Access denied for user 'user2'@'localhost' (using password: YES)
```

没有这个账号了

3.6 修改用户密码

3.6.1 修改管理员密码

`mysqladmin -uroot -p password 新密码 -- 新密码不需要加上引号`

> 注意：需要在未登陆MySQL的情况下操作。

具体操作：

```
mysqladmin -uroot -p password 123456
输入老密码
```

```
                                        新密码
C:\Users\zp>mysqladmin -uroot -p password 123456
Enter password: **** 输入老密码

C:\Users\zp>mysql -uroot -p123456 使用新密码登录
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 24
Server version: 5.5.49 MySQL Community Server (GPL)

Copyright (c) 2000, 2016, Oracle and/or its affiliates. All rights reserved.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql> 登录成功
```

3.6.2 修改普通用户密码

`set password for '用户名'@'主机名' = password('新密码');`

> 注意：需要在登陆MySQL的情况下操作。

具体操作：

```
`set password for 'user1'@'localhost' = password('666666');`
```

```
在MySQL登录的状态下                                    新密码
mysql> set password for `user1`@`localhost` = password('666666');
Query OK, 0 rows affected (0.00 sec)
C:\Users\zp>mysql -uuser1 -p123 老密码登录不了了
ERROR 1045 (28000): Access denied for user 'user1'@'localhost' (using password: YES)

C:\Users\zp>mysql -uuser1 -p666666 使用新密码登录成功了
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 26
Server version: 5.5.49 MySQL Community Server (GPL)

Copyright (c) 2000, 2016, Oracle and/or its affiliates. All rights reserved.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql> _
```

第4章 数据库约束

对表中的数据进行进一步的限制，保证数据的**正确性**、**有效性**和**完整性**。 **约束种类**：

- `PRIMARY KEY`: 主键
- `UNIQUE`: 唯一
- `NOT NULL`: 非空
- `DEFAULT`: 默认
- `FOREIGN KEY`: 外键

4.1主键

4.1.1 主键的作用

用来唯一标识一条记录，每个表都应该有一个主键，并且每个表只能有一个主键。 有些记录的 name,age,score 字段的值都一样时,那么就没法区分这些数据,造成数据库的记录不唯一,这样就不方便管理数据

NAME	age	score
张三	20	80
李四	21	90
王五	19	70
张三	20	80

id	NAME	age	score
1	张三	20	80
2	李四	21	90
3	王五	19	70
4	张三	20	80

哪个字段应该作为表的主键？ 通常不用业务字段作为主键，单独给每张表设计一个id的字段，把id作为主键。主键是给数据库和程序使用的，不是给最终的客户使用的。所以主键有没有含义没有关系，只要不重复，非空就行。

4.1.2 创建主键

主键：`PRIMARY KEY` **主键的特点**：

- 主键必须包含唯一的值
- 主键列不能包含NULL值

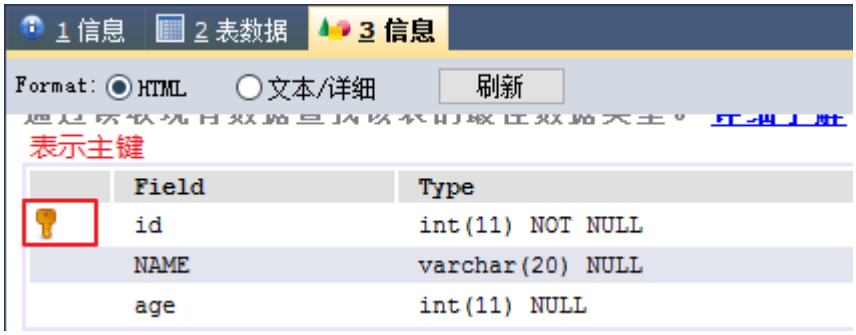
创建主键方式：

1. **在创建表的时候给字段添加主键**
   `字段名 字段类型 PRIMARY KEY`
2. **在已有表中添加主键**
   `ALTER TABLE 表名 ADD PRIMARY KEY(字段名);`

具体操作：

- 创建表学生表st5, 包含字段(id, name, age)将id做为主键

```
CREATE TABLE st5 (
    id INT PRIMARY KEY, -- id是主键
    NAME VARCHAR(20),
    age INT
);
```



- 添加数据

```
INSERT INTO st5 (id, NAME) VALUES (1, '唐伯虎');
INSERT INTO st5 (id, NAME) VALUES (2, '周文宾');
INSERT INTO st5 (id, NAME) VALUES (3, '祝枝山');
INSERT INTO st5 (id, NAME) VALUES (4, '文征明');
```

- 插入重复的主键值

```
-- 主键是唯一的不能重复：Duplicate entry '1' for key 'PRIMARY'
INSERT INTO st5 (id, NAME) VALUES (1, '文征明2');
```

- 插入NULL的主键值

```
-- 主键是不能为空的：Column 'id' cannot be null
INSERT INTO st5 (id, NAME) VALUES (NULL, '文征明3');
```

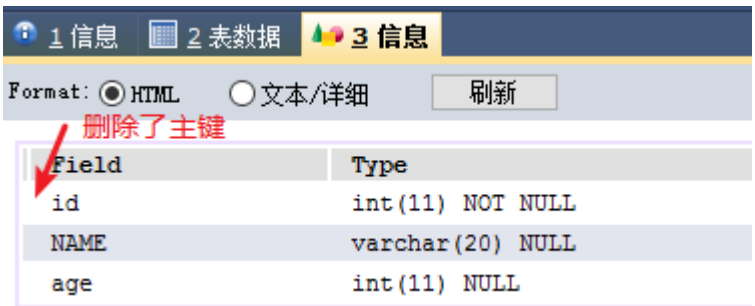
4.1.3 删除主键

`ALTER TABLE 表名 DROP PRIMARY KEY;`

具体操作：

- 删除st5表的主键

```
ALTER TABLE st5 DROP PRIMARY KEY;
```



4.1.4 主键自增

主键如果让我们自己添加很有可能重复,我们通常希望在每次插入新记录时,数据库自动生成主键字段的值 `AUTO_INCREMENT` 表示自动增长(字段类型必须是整数类型)

具体操作：

- 创建学生表st6, 包含字段(id, name, age)将id做为主键并自动增长

```
CREATE TABLE st6 (
    id INT PRIMARY KEY AUTO_INCREMENT,
    NAME VARCHAR(20),
    age INT
);
```

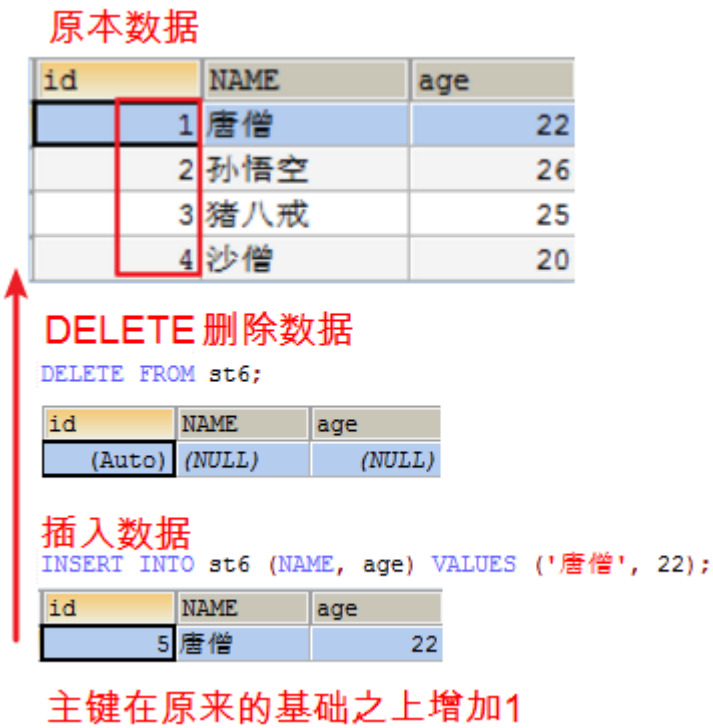
- 插入数据

```
-- 主键默认从1开始自动增长
INSERT INTO st6 (NAME, age) VALUES ('唐僧', 22);
INSERT INTO st6 (NAME, age) VALUES ('孙悟空', 26);
INSERT INTO st6 (NAME, age) VALUES ('猪八戒', 25);
INSERT INTO st6 (NAME, age) VALUES ('沙僧', 20);
```

扩展 默认地AUTO_INCREMENT 的开始值是1，如果希望修改起始值,请使用下列SQL语法 `ALTER TABLE 表名 AUTO_INCREMENT=起始值;`

DELETE和TRUNCATE的区别

- DELETE 删除表中的数据，但不重置AUTO_INCREMENT的值。



- TRUNCATE 摧毁表，重建表，AUTO_INCREMENT重置为1

原本数据

id	NAME	age
1	唐僧	22
2	孙悟空	26
3	猪八戒	25
4	沙僧	20

TRUNCATE摧毁表重建

```
TRUNCATE st6;
```

id	NAME	age
(Auto)	(NULL)	(NULL)

插入数据

```
INSERT INTO st6 (NAME, age) VALUES ('唐僧', 22);
```

id	NAME	age
5	唐僧	22

id	NAME	age
1	唐僧	22

TRUNCATE 摧毁表，重建表，AUTO_INCREMENT重置为1

4.2 唯一

在这张表中这个字段的值不能重复

4.2.1 唯一约束的基本格式

`字段名 字段类型 UNIQUE`

4.2.2 实现唯一约束

具体步骤：

- 创建学生表st7, 包含字段(id, name),name这一列设置唯一约束,不能出现同名的学生

```
CREATE TABLE st7 (
    id INT,
    NAME VARCHAR(20) UNIQUE
);
```

- 添加一个学生

```sql
INSERT INTO st7 VALUES (1, '貂蝉');
INSERT INTO st7 VALUES (2, '西施');
INSERT INTO st7 VALUES (3, '王昭君');
INSERT INTO st7 VALUES (4, '杨玉环');

-- 插入相同的名字出现name重复: Duplicate entry '貂蝉' for key 'name'
INSERT INTO st7 VALUES (5, '貂蝉');

-- 出现多个null的时候会怎样？因为null是没有值，所以不存在重复的问题
INSERT INTO st3 VALUES (5, NULL);
INSERT INTO st3 VALUES (6, NULL);
```

4.3 非空

这个字段必须设置值,不能是NULL

4.3.1 非空约束的基本语法格式

`字段名 字段类型 NOT NULL`

具体操作：

- 创建表学生表st8, 包含字段(id,name,gender)其中name不能为NULL

```sql
CREATE TABLE st8 (
    id INT,
    NAME VARCHAR(20) NOT NULL,
    gender CHAR(2)
);
```

- 添加一条完整的记录

```sql
INSERT INTO st8 VALUES (1, '郭富城', '男');
INSERT INTO st8 VALUES (2, '黎明', '男');
INSERT INTO st8 VALUES (3, '张学友', '男');
INSERT INTO st8 VALUES (4, '刘德华', '男');

-- 姓名不赋值出现姓名不能为null: Column 'name' cannot be null
INSERT INTO st8 VALUES (5, NULL, '男');
```

4.3.2 默认值

往表中添加数据时,如果不指定这个字段的数据,就使用默认值

默认值格式 `字段名 字段类型 DEFAULT 默认值`

具体步骤：

- 创建一个学生表 st9，包含字段(id,name,address)， 地址默认值是广州

```
CREATE TABLE st9 (
    id INT,
    NAME VARCHAR(20),
    address VARCHAR(50) DEFAULT '广州'
);
```

- 添加一条记录,使用默认地址

```
INSERT INTO st9 (id, NAME) VALUES (1, '刘德华');
```

添加一条数据，address不指定数据，
会使用创建表时指定的默认值

```
INSERT INTO st9 (id, NAME) VALUES (1, '刘德华');
```

	id	NAME	address
	1	刘德华	广州
*	(NULL)	(NULL)	广州

从这里也可以看出表的默认值

- 添加一条记录,不使用默认地址

```
INSERT INTO st9 VALUES (2, '张学友', '香港');
```

疑问:如果一个字段设置了非空与唯一约束，该字段与主键的区别

- 一张表中只有一个字段可以设置为主键
- 一张表中可以多个字段非空与唯一约束
- 主键可以自动增长，非空与唯一约束的字段不能自动增长

4.4 外键

4.4.1 单表的缺点

创建一个员工表包含如下列(id, name, age, dep_name, dep_location),id主键并自动增长,添加5条数据

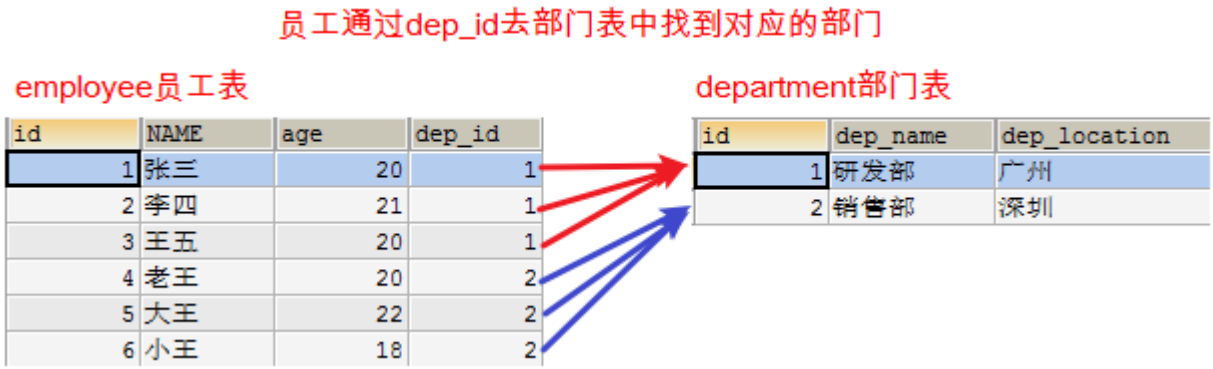
```
CREATE TABLE emp (
    id INT PRIMARY KEY AUTO_INCREMENT,
    NAME VARCHAR(30),
    age INT,
    dep_name VARCHAR(30),
    dep_location VARCHAR(30)
);

-- 添加数据
INSERT INTO emp (NAME, age, dep_name, dep_location) VALUES ('张三', 20, '研发部', '广州');
INSERT INTO emp (NAME, age, dep_name, dep_location) VALUES ('李四', 21, '研发部', '广州');
INSERT INTO emp (NAME, age, dep_name, dep_location) VALUES ('王五', 20, '研发部', '广州');
```

```sql
INSERT INTO emp (NAME, age, dep_name, dep_location) VALUES ('老王', 20, '销售部', '深圳');
INSERT INTO emp (NAME, age, dep_name, dep_location) VALUES ('大王', 22, '销售部', '深圳');
INSERT INTO emp (NAME, age, dep_name, dep_location) VALUES ('小王', 18, '销售部', '深圳');
```

缺点:表中出现了很多重复的数据(数据冗余)，如果要修改研发部的地址需要修改3个地方。

id	name	age	dep_name	dep_location
1	张三	20	研发部	广州
2	李四	21	研发部	广州
3	王五	20	研发部	广州
4	老王	20	销售部	深圳
5	大王	22	销售部	深圳
6	小王	18	销售部	深圳

解决方案:将一张表分成2张表(员工表和部门表)




```sql
-- 创建部门表
CREATE TABLE department (
    id INT PRIMARY KEY AUTO_INCREMENT,
    dep_name VARCHAR(20),
    dep_location VARCHAR(20)
);

-- 创建员工表
CREATE TABLE employee (
    id INT PRIMARY KEY AUTO_INCREMENT,
    NAME VARCHAR(20),
    age INT,
    dep_id INT
);

-- 添加2个部门
INSERT INTO department (dep_name, dep_location) VALUES ('研发部', '广州'), ('销售部', '深圳');

-- 添加员工,dep_id表示员工所在的部门
INSERT INTO employee (NAME, age, dep_id) VALUES
('张三', 20, 1),
('李四', 21, 1),
('王五', 20, 1),
('老王', 20, 2),
('大王', 22, 2),
('小王', 18, 2);
```

问题： 当我们在employee的dep_id里面输入不存在的部门,数据依然可以添加.但是并没有对应的部门，不能出现这种情况。employee的dep_id中的内容只能是department表中存在的id

id	NAME	age	dep_id
1	张三	20	1
2	李四	21	1
3	王五	20	1
4	老王	20	2
5	大王	22	2
6	小王	18	2
7	老张	18	6

这条记录的
dep_id有问题

id	dep_name	dep_location
1	研发部	广州
2	销售部	深圳

目标:需要约束dep_id只能是department表中已经存在id **解决方式**:使用外键约束

4.4.2 什么是外键约束

一张表中的某个字段引用另一个表的主键 主表： 约束别人 副表/从表： 使用别人的数据，被别人约束

一张表中的某个字段引用另一个表的主键

employee员工表　　外键

id	NAME	age	dep_id
1	张三	20	1
2	李四	21	1
3	王五	20	1
4	老王	20	2
5	大王	22	2
6	小王	18	2

副表/从表： 使用别人的数据，被别人约束

主键　　department部门表

id	dep_name	dep_location
1	研发部	广州
2	销售部	深圳

主表： 约束别人

4.4.3 创建外键

1. **新建表时增加外键**：
   `[CONSTRAINT] [外键约束名称] FOREIGN KEY(外键字段名) REFERENCES 主表名(主键字段名)`
   关键字解释：
   CONSTRAINT -- 约束关键字
   FOREIGN KEY(外键字段名) -- 某个字段作为外键
   REFERENCES -- 主表名(主键字段名) 表示参照主表中的某个字段
2. **已有表增加外键**：
   `ALTER TABLE 从表 ADD [CONSTRAINT] [外键约束名称] FOREIGN KEY (外键字段名) REFERENCES 主表(主键字段名);`

具体操作：

- 副表/从表: 被别人约束,表结构添加外键约束
- 删除副表/从表 employee
- 创建从表 employee 并添加外键约束

```sql
CREATE TABLE employee (
    id INT PRIMARY KEY AUTO_INCREMENT,
    NAME VARCHAR(20),
    age INT,
    dep_id INT,
    -- 添加一个外键
    -- 外键取名公司要求,一般fk结尾
    CONSTRAINT emp_depid_ref_dep_id_fk FOREIGN KEY(dep_id) REFERENCES department(id)
);
```

- 正常添加数据

```sql
INSERT INTO employee (NAME, age, dep_id) VALUES
('张三', 20, 1),
('李四', 21, 1),
('王五', 20, 1),
('老王', 20, 2),
('大王', 22, 2),
('小王', 18, 2);
```

- 部门错误的数据添加失败

```sql
INSERT INTO employee (NAME, age, dep_id) VALUES ('二王', 20, 5);
```

4.4.4 删除外键

`ALTER TABLE 从表 drop foreign key 外键名称;`

具体操作：

- 删除employee表的emp_depid_ref_dep_id_fk外键

```sql
ALTER TABLE employee DROP FOREIGN KEY emp_depid_ref_dep_id_fk;
```

- 在employee表情存在况下添加外键

```sql
ALTER TABLE employee ADD CONSTRAINT emp_depid_ref_dep_id_fk FOREIGN KEY(dep_id) REFERENCES
department(id);
```

4.4.5 外键的级联

要把部门表中的id值2，改成5，能不能直接修改呢？

```sql
UPDATE department SET id=5 WHERE id=2;
```

不能直接修改:Cannot delete or update a parent row: a foreign key constraint fails 如果副表(员工表)中有引用的数据,不能直接修改主表(部门表)主键

要删除部门id等于1的部门, 能不能直接删除呢？

```
UPDATE department SET id=5 WHERE id=2;
```

不能直接删除:Cannot delete or update a parent row: a foreign key constraint fails 如果副表(员工表)中有引用的数据,不能直接删除主表(部门表)数据

什么是级联操作： 在修改和删除主表的主键时，同时更新或删除副表的外键值，称为级联操作 `ON UPDATE CASCADE` -- 级联更新，主键发生更新时，外键也会更新 `ON DELETE CASCADE` -- 级联删除，主键发生删除时，外键也会删除

具体操作：

- 删除employee表
- 重新创建employee表，添加级联更新和级联删除

```
CREATE TABLE employee (
    id INT PRIMARY KEY AUTO_INCREMENT,
    NAME VARCHAR(30),
    age INT,
    dep_id INT,
    -- 添加外键约束,并且添加级联更新和级联删除
    CONSTRAINT employee_dep_fk FOREIGN KEY (dep_id) REFERENCES department(id) ON UPDATE CASCADE
ON DELETE CASCADE
);
```

- 再次添加数据到员工表和部门表

```
INSERT INTO employee (NAME, age, dep_id) VALUES ('张三', 20, 1);
INSERT INTO employee (NAME, age, dep_id) VALUES ('李四', 21, 1);
INSERT INTO employee (NAME, age, dep_id) VALUES ('王五', 20, 1);
INSERT INTO employee (NAME, age, dep_id) VALUES ('老王', 20, 2);
INSERT INTO employee (NAME, age, dep_id) VALUES ('大王', 22, 2);
INSERT INTO employee (NAME, age, dep_id) VALUES ('小王', 18, 2);
```

- 把部门表中id等于1的部门改成id等于10

```
UPDATE department SET id=10 WHERE id=1;
```

id	NAME	age	dep_id
1	张三	20	10
2	李四	21	10
3	王五	20	10
7	老王	20	2
8	大王	22	2
9	小王	18	2

id	dep_name	dep_location
2	销售部	深圳
10	研发部	广州

- 删除部门号是2的部门

```
DELETE FROM department WHERE id=2;
```

id	NAME	age	dep_id
1	张三	20	10
2	李四	21	10
3	王五	20	10

id	dep_name	dep_location
10	研发部	广州

第5章 三范式

5.1 什么是范式

范式是指:设计数据库表的规则(Normal Form) 好的数据库设计对数据的存储性能和后期的程序开发，都会产生重要的影响。建立科学的，规范的数据库就需要满足一些规则来优化数据的设计和存储

5.2 范式的基本分类

目前关系数据库有六种范式：第一范式（1NF）、第二范式（2NF）、第三范式（3NF）、巴斯-科德范式（BCNF）、第四范式(4NF）和第五范式（5NF，又称完美范式）。满足最低要求的范式是第一范式（1NF）。在第一范式的基础上进一步满足更多规范要求的称为第二范式（2NF），其余范式以次类推。**一般说来，数据库只需满足第三范式(3NF）就行了。**

5.3 第一范式

即数据库表的每一列都是不可分割的原子数据项，而不能是集合、数组、记录等非原子数据项。即实体中的某个属性有多个值时，必须拆分为不同的属性。在符合第一范式（1NF）表中每个列的值只能是表的一个属性或一个属性的一部分。简而言之，第一范式每一列不可再拆分，称为原子性。 **第一范式**：每一列不能再拆分

比如上课时间和下课时间

姓名	性别	班级名称	教室	考勤时间(开始,结束)
老王	男	JavaEE	110	18-01-10 8:10, 18-01-10 21:30
张三	女	UI设计	120	18-01-10 8:11, 18-01-10 21:33
小王	男	JavaEE	110	18-01-11 8:12, 18-01-11 21:36
张三	女	UI设计	120	18-01-11 8:12, 18-01-11 21:35

满足第一范式

姓名	性别	班级名称	教室	开始考勤时间	结束考勤时间
老王	男	JavaEE	110	2018/1/10 8:10	2018/1/10 21:30
张三	女	UI设计	120	2018/1/10 8:11	2018/1/10 21:33
小王	男	JavaEE	110	2018/1/11 8:12	,18-01-11 21:36
张三	女	UI设计	120	2018/1/11 8:12	2018/1/11 21:35

> 总结：如果不遵守第一范式，查询出数据还需要进一步处理（查询不方便）。遵守第一范式，需要什么字段的数据就查询什么数据（方便查询）。

5.4 第二范式

第二范式（2NF）要求数据库表中的每个实例或记录必须可以被唯一地区分。选取一个能区分每个实体的属性或属性组，作为实体的唯一标识。例如在员工表中的身份证号码即可实现每个员工的区分，该身份证号码即为候选键，任何一个候选键都可以被选作主键。在找不到候选键时，可额外增加属性以实现区分。 第二范式（2NF）要求实体的属性完全依赖于主关键字。所谓完全依赖是指不能存在仅依赖主关键字一部分的属性。如果存在，那么这个属性和主关键字的这一部分应该分离出来形成一个新的实体，新实体与原实体之间是一对多的关系。为实现区分通常需要为表加上一个列，以存储各个实例的唯一标识。简而言之，第二范式就是在第一范式的基础上属性完全依赖于主键。

第二范式：

1. 一张表只描述一件事情
2. 表中的每一个字段都依赖于主键

姓名	性别	班级名称	教室	开始考勤时间	结束考勤时间
老王	男	JavaEE	110	2018/1/10 8:10	2018/1/10 21:30
张三	女	UI设计	120	2018/1/10 8:11	2018/1/10 21:33
小王	男	JavaEE	110	2018/1/11 8:12	,18-01-11 21:36
张三	女	UI设计	120	2018/1/11 8:12	2018/1/11 21:35

第二范式:
一张表只描述一件事情
表中的每一个字段都依赖于主键

一张表只描述一件事情，分成3张表

学生表

姓名	性别
老王	男
张三	女
小王	男
张三	女

班级表

班级名称	教室
JavaEE	110
UI设计	120

考勤时间表

开始考勤时间	结束考勤时间
2018/1/10 8:10	2018/1/10 21:30
2018/1/10 8:11	2018/1/10 21:33
2018/1/11 8:12	,18-01-11 21:36
2018/1/11 8:12	2018/1/11 21:35

姓名	性别	班级名称	教室	开始考勤时间	结束考勤时间
老王	男	JavaEE	110	2018/1/10 8:10	2018/1/10 21:30
张三	女	UI设计	120	2018/1/10 8:11	2018/1/10 21:33
小王	男	JavaEE	110	2018/1/11 8:12	,18-01-11 21:36
张三	女	UI设计	120	2018/1/11 8:12	2018/1/11 21:35

一张表只描述一件事情，分成3张表

学生表

姓名	性别
老王	男
张三	女
小王	男
张三	女

班级表

班级名称	教室
JavaEE	110
UI设计	120

考勤时间表

开始考勤时间	结束考勤时间
2018/1/10 8:10	2018/1/10 21:30
2018/1/10 8:11	2018/1/10 21:33
2018/1/11 8:12	,18-01-11 21:36
2018/1/11 8:12	2018/1/11 21:35

表中的每一个字段都依赖于主键

学生表

学号	姓名	性别
1	老王	男
2	张三	女
3	小王	男
4	张三	女

班级表

班级号	班级名称	教室
1	JavaEE	110
2	UI设计	120

考勤时间表

考勤号	开始考勤时间	结束考勤时间
1	2018/1/10 8:10	2018/1/10 21:30
2	2018/1/10 8:11	2018/1/10 21:33
3	2018/1/11 8:12	,18-01-11 21:36
4	2018/1/11 8:12	2018/1/11 21:35

> 总结：如果不准守第二范式，数据冗余，相同数据无法区分。遵守第二范式减少数据冗余，通过主键区分相同数据。

5.5 第三范式

在2NF基础上，任何非主属性不依赖于其它非主属性（在2NF基础上消除传递依赖） 第三范式（3NF）是第二范式（2NF）的一个子集，即满足第三范式（3NF）必须满足第二范式（2NF）。简而言之，第三范式（3NF）要求一个关系中不包含已在其它关系已包含的非主关键字信息。

第三范式：引用其他表的主键

第三范式：任何非主属性不依赖于其它非主属性

不满足第三范式，班级名称引用的是班级表中的非主属性

id	学号	班级名称	考勤时间号
1	1	JavaEE	1
2	2	UI设计	2
3	3	JavaEE	3
4	4	UI设计	4

姓名	性别	班级名称	教室	开始考勤时间	结束考勤时间
老王	男	JavaEE	110	2018/1/10 8:10	2018/1/10 21:30
张三	女	UI设计	120	2018/1/10 8:11	2018/1/10 21:33
小王	男	JavaEE	110	2018/1/11 8:12	,18-01-11 21:36
张三	女	UI设计	120	2018/1/11 8:12	2018/1/11 21:35

满足第三范式，非主属性学号，班级号，考勤时间号都是引用其他表的主属性

id	学号	班级号	考勤时间号
1	1	1	1
2	2	2	2
3	3	1	3
4	4	2	4

> 总结：如果不准守第三范式，可能会有相同数据无法区分，修改数据的时候多张表都需要修改（不方便修改）。遵守第三范式通过id可以区分相同数据，修改数据的时候只需要修改一张表（方便修改）。

第6章 表关系

6.1 表关系的概念

现实生活中，实体与实体之间肯定是有关系的，比如：老公和老婆，部门和员工，老师和学生等。那么我们在设计表的时候，就应该体现出表与表之间的这种关系！分成三种：

1. 一对一
2. 一对多
3. 多对多

6.2 一对多

一对多（1:n） 例如：班级和学生，部门和员工，客户和订单，分类和商品 一对多建表原则: 在从表(多方)创建一个字段,字段作为外键指向主表(一方)的主键

一对多关系 一个部门有多个员工

id	NAME	age
1	张三	20
2	李四	21
3	王五	20
4	老王	20
5	大王	22
6	小王	18

id	dep_name	dep_location
1	研发部	广州
2	销售部	深圳

一对多关系 一个部门有多个员工

多方

id	NAME	age	dep_id
1	张三	20	1
2	李四	21	1
3	王五	20	1
4	老王	20	2
5	大王	22	2
6	小王	18	2

一方

id	dep_name	dep_location
1	研发部	广州
2	销售部	深圳

6.3 多对多

多对多（m:n）例如：老师和学生，学生和课程，用户和角色 多对多关系建表原则: 需要创建第三张表，中间表中至少两个字段，这两个字段分别作为外键指向各自一方的主键。

张三选择语文和数学

李四选择数学和英语

多对多

学生表

学号	姓名
1	张三
2	李四
3	王五

中间表

学生-课程关系表

学号	课程号
1	1
1	2
2	2
2	3
3	3

课程表

课程号	课程名
1	语文
2	数学
3	英语

6.4 一对一

一对一（1:1）在实际的开发中应用不多.因为一对一可以创建成一张表。 两种建表原则：

- 外键唯一：主表的主键和从表的外键（唯一），形成主外键关系，外键唯一`UNIQUE`

- 外键是主键：主表的主键和从表的主键，形成主外键关系

一对一

学生表

学号	姓名	简历号
1	张三	1
2	李四	2
3	王五	3

简历表

简历号	简历
1	张三的简历
2	李四的简历
3	王五的简历

一对一

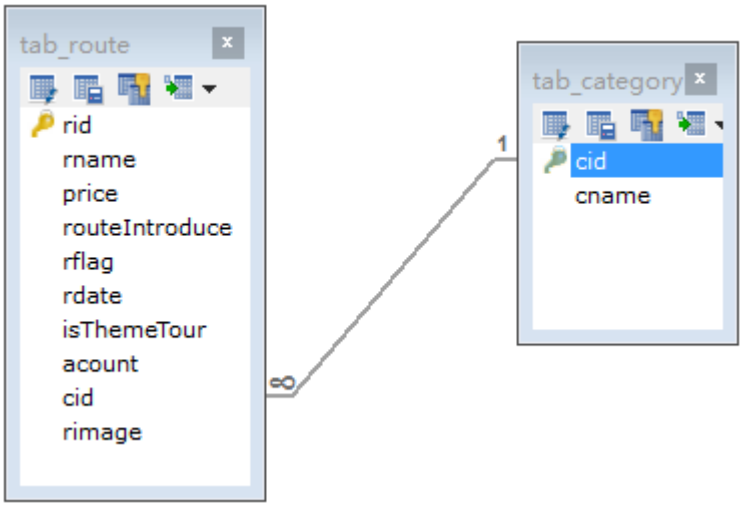
学生表

学号	姓名
1	张三
2	李四
3	王五

个人信息

编号	出生地	曾用名	出生体重
1	广东	四毛	6.2
2	广西	三毛	5.5
3	江西	小王	7.3

一对多关系练习 一个旅游线路分类中有多个旅游线路



一个旅游线路分类中有多个旅游线路，属于一对多

多方

rid	rname	price	routeIntroduce	rflag	rdate	isTh...	ac...	cid	rimage
1	【厦门+鼓浪屿+南普陀寺+曾厝垵 高铁3天 惠费团】尝	1499	春节国内游优惠：即日起至2018年1月31号，5-9人	1	2018-01-27	0	100	3	c:1.png
2	【浪漫桂林 阳朔西街高铁3天纯玩 高级团】城徽象鼻山	699	春节国内游优惠：即日起至2018年1月31号，5-9人	1	2018-01-27	0	66	3	c:2.png
3	【爆款￥1699秒杀】泰国 曼谷 芭堤雅 金沙岛 杜拉拉	1699	1月15日至2月11日官网特卖！①出境全线正价线路	1	2018-01-27	0	15	2	c:123.png
4	【经典·狮航 ￥2399秒杀】巴厘岛双飞五天 抵玩【广州	2399	官网特卖！2-3月出发，前10名网付立享￥2399/人	1	2018-01-27	0	22	2	c:3.png
5	香港迪士尼乐园自由行2天【永东跨境巴士广东至迪士尼	799	永东巴士提供广东省内多个上车地点，购买后需自	1	2018-01-27	0	38	4	c:4.png

1方

cid	cname
1	周边游
2	出境游
3	国内游
4	港澳游
(Auto)	(NULL)

具体操作：

- 创建旅游线路分类表

```
CREATE TABLE tab_category (
   cid INT PRIMARY KEY AUTO_INCREMENT, -- 旅游线路分类主键
   cname VARCHAR(100) NOT NULL UNIQUE -- 旅游线路分类名称
);
```

- 添加旅游线路分类数据

```
INSERT INTO tab_category (cname) VALUES ('周边游'), ('出境游'), ('国内游'), ('港澳游');
```

- 创建旅游线路表

```
CREATE TABLE tab_route (
   rid INT PRIMARY KEY AUTO_INCREMENT, -- 旅游线路主键
   rname VARCHAR(100) NOT NULL UNIQUE, -- 旅游线路名称
   price DOUBLE NOT NULL,  -- 价格
   routeIntroduce VARCHAR(200), -- 线路介绍
   rflag CHAR(1) NOT NULL, -- 是否上架
   rdate VARCHAR(19) NOT NULL, -- 上架时间
   isThemeTour CHAR(1) NOT NULL, -- 是否主题旅游
   acount INT DEFAULT 0, -- 收藏数量
   cid INT NOT NULL, -- 所属分类
   rimage VARCHAR(200) NOT NULL, -- 缩略图地址
   CONSTRAINT ro_cid_ref_cate_id FOREIGN KEY(cid) REFERENCES tab_category(cid)
);
```

- 添加旅游线路数据

```
INSERT INTO tab_route VALUES
(NULL, '【厦门+鼓浪屿+南普陀寺+曾厝垵 高铁3天 惠贵团】尝味友鸭面线 住1晚鼓浪屿', 1499, '春节国内游优惠：即日起至2018年1月31号，5-9人同时报名立减￥100/人，10人及以上同时报名立减￥150/人。仅限2月10-22日春节期间出发享优惠，自由行及特价产品不参与！', 1, '2018-01-27', 0, 100, 3, 'c:\1.png'),
(NULL, '【浪漫桂林 阳朔西街高铁3天纯玩 高级团】城徽象鼻山 兴坪漓江 西山公园', 699, '春节国内游优惠：即日起至2018年1月31号，5-9人同时报名立减￥100/人，10人及以上同时报名立减￥150/人。仅限2月10-22日春节期间出发享优惠，自由行及特价产品不参与！', 1, '2018-01-27', 0, 66, 3, 'c:\2.png'),
(NULL, '【爆款￥1699秒杀】泰国 曼谷 芭堤雅 金沙岛 杜拉拉水上市场 双飞六天【含送签费 泰风情 广州往返 特价团】', 1699, '1月15日至2月11日官网特卖！①出境全线正价线路（特价线除外）满2人立减￥60！满4人立减￥200！满5人立减￥500！', 1, '2018-01-27', 0, 15, 2, 'c:\123.png'),
(NULL, '【经典·狮航 ￥2399秒杀】巴厘岛双飞五天 抵玩【广州往返 特价团】', 2399, '官网特卖！2-3月出发，前10名网付立享￥2399/人！', 1, '2018-01-27', 0, 22, 2, 'c:\3.png'),
(NULL, '香港迪士尼乐园自由行2天【永东跨境巴士广东至迪士尼去程交通+迪士尼一日门票+香港如心海景酒店暨会议中心标准房1晚住宿】', 799, '永东巴士提供广东省内多个上车地点，购买后需自行致电永东巴士客服电话4008861668预约车位', 1, '2018-01-27', 0, 38, 4, 'c:\4.png');
```

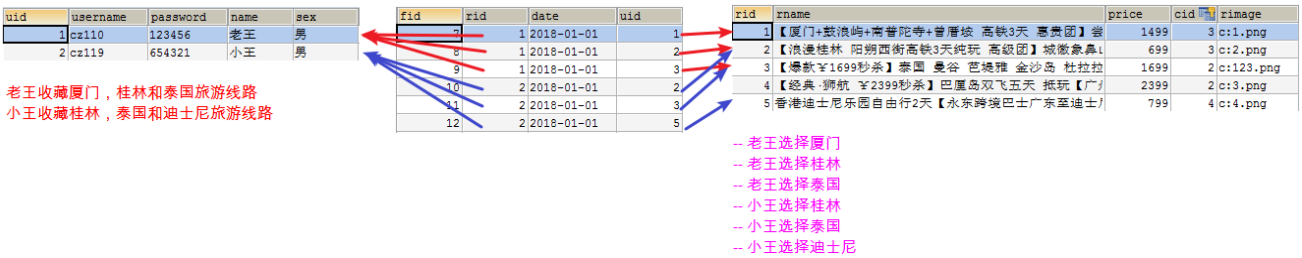
多对多关系练习 一个用户收藏多个线路，一个线路被多个用户收藏

uid	username	password	name	sex
1	cz110	123456	老王	男
2	cz119	654321	小王	男

rid	rname	price	cid	rimage
1	【厦门+鼓浪屿+南普陀寺+曾厝垵 高铁3天 惠贵团】尝	1499	3	c:1.png
2	【浪漫桂林 阳朔西街高铁3天纯玩 高级团】城徽象鼻山	699	3	c:2.png
3	【爆款￥1699秒杀】泰国 曼谷 芭堤雅 金沙岛 杜拉拉	1699	2	c:123.png
4	【经典·狮航 ￥2399秒杀】巴厘岛双飞五天 抵玩【广	2399	2	c:3.png
5	香港迪士尼乐园自由行2天【永东跨境巴士广东至迪士尼	799	4	c:4.png

老王收藏厦门，桂林和泰国旅游线路
小王收藏桂林，泰国和迪士尼旅游线路

对于多对多的关系我们需要增加一张中间表来维护他们之间个关系



具体操作：

- 创建用户表

```
CREATE TABLE tab_user (
   uid INT PRIMARY KEY AUTO_INCREMENT, -- 用户id
   username VARCHAR(100) NOT NULL UNIQUE, -- 用户名
   PASSWORD VARCHAR(30) NOT NULL, -- 密码
   NAME VARCHAR(100), -- 真实姓名
   birthday DATE, -- 生日
   sex CHAR(1), -- 性别
   telephone VARCHAR(11), -- 手机号
   email VARCHAR(100), -- 邮箱
   STATUS CHAR(1) NOT NULL, -- 是否激活状态
   CODE VARCHAR(32) NOT NULL UNIQUE -- 激活码
);
```

- 添加用户数据

```
INSERT INTO tab_user VALUES
(NULL, 'cz110', 123456, '老王', '1977-07-07', '男', '13888888888', '66666@qq.com', '是', '1386'),
(NULL, 'cz119', 654321, '小王', '1999-09-09', '男', '13999999999', '99999@qq.com', '是', '9898');
```

- 创建收藏表

```sql
CREATE TABLE tab_favorite (
   fid INT PRIMARY KEY AUTO_INCREMENT, -- 收藏主键
   rid INT NOT NULL, -- 旅游线路id
   DATE DATE NOT NULL, -- 收藏时间
   uid INT NOT NULL -- 用户id
);
```

- 增加收藏表数据

```sql
INSERT INTO tab_favorite VALUES
(NULL, 1, '2018-01-01', 1), -- 老王选择厦门
(NULL, 1, '2018-01-01', 2), -- 老王选择桂林
(NULL, 1, '2018-01-01', 3), -- 老王选择泰国
(NULL, 2, '2018-01-01', 2), -- 小王选择桂林
(NULL, 2, '2018-01-01', 3), -- 小王选择泰国
(NULL, 2, '2018-01-01', 5); -- 小王选择迪士尼
```