

1.2 语法

DQL查询语句，语法结构如下：

```
1  SELECT
2        字段列表
3  FROM
4        表名列列表
5  WHERE
6        条件列表
7  GROUP BY
8        分组字段列表
9  HAVING
10       分组后条件列表
11 ORDER BY
12       排序字段列表
13 LIMIT
14       分页参数
```

我们今天会将上面的完整语法拆分为以下几个部分学习：

- 基本查询（不带任何条件）
- 条件查询（where）
- 分组查询（group by）
- 排序查询（order by）
- 分页查询（limit）

准备一些测试数据用于查询操作：

```
1  create database db02; -- 创建数据库
2  use db02; -- 切换数据库
3  -- 员工管理(带约束)
4  create table tb_emp (
5      id int unsigned primary key auto_increment comment 'ID',
6      username varchar(20) not null unique comment '用户名',
7      password varchar(32) default '123456' comment '密码',
8      name varchar(10) not null comment '姓名',
9      gender tinyint unsigned not null comment '性别，说明：1 男，2 女',
10     image varchar(300) comment '图像',
11     job tinyint unsigned comment '职位，说明：1 班主任,2 讲师, 3 学工主管, 4 教研主管',
12     entrydate date comment '入职时间',
13     create_time datetime not null comment '创建时间',
14     update_time datetime not null comment '修改时间'
```

```
15 ) comment '员工表';
16
17 -- 准备测试数据
18 INSERT INTO tb_emp (id, username, password, name, gender, image,
19 job, entrydate, create_time, update_time) VALUES
20 (1, 'jinyong', '123456', '金庸', 1, '1.jpg', 4, '2000-01-01',
21 '2022-10-27 16:35:33', '2022-10-27 16:35:35'),
22 (2, 'zhangwuji', '123456', '张无忌', 1, '2.jpg', 2, '2015-01-01',
23 '2022-10-27 16:35:33', '2022-10-27 16:35:37'),
24 (3, 'yangxiao', '123456', '杨逍', 1, '3.jpg', 2, '2008-05-01',
25 '2022-10-27 16:35:33', '2022-10-27 16:35:39'),
26 (4, 'weiyixiao', '123456', '韦一笑', 1, '4.jpg', 2, '2007-01-01',
27 '2022-10-27 16:35:33', '2022-10-27 16:35:41'),
28 (5, 'changyuchun', '123456', '常遇春', 1, '5.jpg', 2, '2012-12-
29 05', '2022-10-27 16:35:33', '2022-10-27 16:35:43'),
30 (6, 'xiaozhao', '123456', '小昭', 2, '6.jpg', 3, '2013-09-05',
31 '2022-10-27 16:35:33', '2022-10-27 16:35:45'),
32 (7, 'jixiaofu', '123456', '纪晓芙', 2, '7.jpg', 1, '2005-08-01',
33 '2022-10-27 16:35:33', '2022-10-27 16:35:47'),
34 (8, 'zhouzhiruo', '123456', '周芷若', 2, '8.jpg', 1, '2014-11-
35 09', '2022-10-27 16:35:33', '2022-10-27 16:35:49'),
36 (9, 'dingminjun', '123456', '丁敏君', 2, '9.jpg', 1, '2011-03-
37 11', '2022-10-27 16:35:33', '2022-10-27 16:35:51'),
38 (10, 'zhaomin', '123456', '赵敏', 2, '10.jpg', 1, '2013-09-05',
39 '2022-10-27 16:35:33', '2022-10-27 16:35:53'),
40 (11, 'luzhangke', '123456', '鹿杖客', 1, '11.jpg', 2, '2007-02-
41 01', '2022-10-27 16:35:33', '2022-10-27 16:35:55'),
42 (12, 'hebiweng', '123456', '鹤笔翁', 1, '12.jpg', 2, '2008-08-
43 18', '2022-10-27 16:35:33', '2022-10-27 16:35:57'),
44 (13, 'fangdongbai', '123456', '方东白', 1, '13.jpg', 1, '2012-11-
45 01', '2022-10-27 16:35:33', '2022-10-27 16:35:59'),
46 (14, 'zhangsanfeng', '123456', '张三丰', 1, '14.jpg', 2, '2002-
47 08-01', '2022-10-27 16:35:33', '2022-10-27 16:36:01'),
48 (15, 'yulianzhou', '123456', '俞莲舟', 1, '15.jpg', 2, '2011-05-
49 01', '2022-10-27 16:35:33', '2022-10-27 16:36:03'),
50 (16, 'songyuanqiao', '123456', '宋远桥', 1, '16.jpg', 2, '2010-
51 01-01', '2022-10-27 16:35:33', '2022-10-27 16:36:05'),
52 (17, 'chenyouliang', '12345678', '陈友谅', 1, '17.jpg', null,
53 '2015-03-21', '2022-10-27 16:35:33', '2022-10-27 16:36:07'),
54 (18, 'zhang1', '123456', '张一', 1, '2.jpg', 2, '2015-01-01',
55 '2022-10-27 16:35:33', '2022-10-27 16:36:09'),
56 (19, 'zhang2', '123456', '张二', 1, '2.jpg', 2, '2012-01-01',
57 '2022-10-27 16:35:33', '2022-10-27 16:36:11'),
```

```
38      (20, 'zhang3', '123456', '张三', 1, '2.jpg', 2, '2018-01-01',  
      '2022-10-27 16:35:33', '2022-10-27 16:36:13'),  
39      (21, 'zhang4', '123456', '张四', 1, '2.jpg', 2, '2015-01-01',  
      '2022-10-27 16:35:33', '2022-10-27 16:36:15'),  
40      (22, 'zhang5', '123456', '张五', 1, '2.jpg', 2, '2016-01-01',  
      '2022-10-27 16:35:33', '2022-10-27 16:36:17'),  
41      (23, 'zhang6', '123456', '张六', 1, '2.jpg', 2, '2012-01-01',  
      '2022-10-27 16:35:33', '2022-10-27 16:36:19'),  
42      (24, 'zhang7', '123456', '张七', 1, '2.jpg', 2, '2006-01-01',  
      '2022-10-27 16:35:33', '2022-10-27 16:36:21'),  
43      (25, 'zhang8', '123456', '张八', 1, '2.jpg', 2, '2002-01-01',  
      '2022-10-27 16:35:33', '2022-10-27 16:36:23'),  
44      (26, 'zhang9', '123456', '张九', 1, '2.jpg', 2, '2011-01-01',  
      '2022-10-27 16:35:33', '2022-10-27 16:36:25'),  
45      (27, 'zhang10', '123456', '张十', 1, '2.jpg', 2, '2004-01-01',  
      '2022-10-27 16:35:33', '2022-10-27 16:36:27'),  
46      (28, 'zhang11', '123456', '张十一', 1, '2.jpg', 2, '2007-01-01',  
      '2022-10-27 16:35:33', '2022-10-27 16:36:29'),  
47      (29, 'zhang12', '123456', '张十二', 1, '2.jpg', 2, '2020-01-01',  
      '2022-10-27 16:35:33', '2022-10-27 16:36:31');
```

1.3 基本查询

在基本查询的DQL语句中，不带任何的查询条件，语法如下：

- 查询多个字段

```
1  select 字段1, 字段2, 字段3 from 表名;
```

- 查询所有字段（通配符）

```
1  select * from 表名;
```

- 设置别名

```
1  select 字段1 [ as 别名1 ], 字段2 [ as 别名2 ] from 表名;
```

- 去除重复记录

```
1  select distinct 字段列表 from 表名;
```

案例1：查询指定字段 name, entrydate并返回

```
1  select name,entrydate from tb_emp;
```

```
49
50 -- 1. 查询指定字段 name,entrydate 并返回
51 ✓ select name,entrydate from tb_emp;
52
53
```

Output Result 21

查询结果

	name	entrydate
1	金庸	2000-01-01
2	张无忌	2015-01-01
3	杨逍	2008-05-01
4	韦一笑	2007-01-01

案例2：查询返回所有字段

```
1 select * from tb_emp;
```

* 号代表查询所有字段，在实际开发中尽量少用（不直观、影响效率）

```
58 -- 2. 查询返回所有字段
59 ✓ select * from tb_emp;
60
```

Output 2. 查询返回所有字段

	id	username	password	name	gender	image	job	entrydate	create_time	update_time
1	1	jinyong	123456	金庸		1.1.jpg	4	2000-01-01	2022-10-27 16:35:33	2022-10-27 16:35
2	2	zhangwuji	123456	张无忌		1.2.jpg	2	2015-01-01	2022-10-27 16:35:33	2022-10-27 16:35
3	3	yangxiao	123456	杨逍		1.3.jpg	2	2008-05-01	2022-10-27 16:35:33	2022-10-27 16:35
4	4	weiyixiao	123456	韦一笑		1.4.jpg	2	2007-01-01	2022-10-27 16:35:33	2022-10-27 16:35

案例3：查询所有员工的 name,entrydate, 并起别名(姓名、入职日期)

```
1 -- 方式1:
2 select name AS 姓名, entrydate AS 入职日期 from tb_emp;
3 -- 方式2: 别名中有特殊字符时, 使用 '' 或 "" 包含
4 select name AS '姓 名', entrydate AS '入职日期' from tb_emp;
5 -- 方式3:
6 select name AS "姓名", entrydate AS "入职日期" from tb_emp;
```

```

59 select * from tb_emp;
60
61 -- 3. 查询所有员工的 name,entrydate, 并起别名(姓名、入职日期)
62 select name AS 姓名, entrydate AS 入职日期 from tb_emp;
63 select name AS '姓 名', entrydate AS '入职日期' from tb_emp;
64 select name AS "姓名", entrydate AS "入职日期" from tb_emp;

```

Output Result 28

29 rows

	姓名	入职日期
1	金庸	2000-01-01
2	张无忌	2015-01-01
3	杨逍	2008-05-01
4	韦一笑	2007-01-01

查询结果，以别名作为字段名显示

案例4：查询已有的员工关联了哪几种职位（不要重复）

```

1 select distinct job from tb_emp;

```

-- 4. 查询已有的员工关联了哪几种职位(不要重复)

```

select job from tb_emp;

```

Output Result 33

29 rows

	job
1	4
2	2
3	2
4	2
5	2
6	3
7	1
8	1
9	1

重复数据

-- 4. 查询已有的员工关联了哪几种职位(不要重复)

```

select distinct job from tb_emp;

```

Output jobtinyint unsigned 2

5 rows

	job
1	4
2	2
3	3
4	1
5	<null>

对查询去重

没有重复数据

1.4 条件查询

语法：

```

1 select 字段列表 from 表名 where 条件列表 ; -- 条件列表：意味着可以有多个条件

```

学习条件查询就是学习条件的构建方式，而在SQL语句当中构造条件的运算符分为两类：

- 比较运算符
- 逻辑运算符

常用的比较运算符如下：

比较运算符	功能
>	大于
>=	大于等于
<	小于
<=	小于等于
=	等于
<> 或 !=	不等于
<u>between ... and ...</u>	<u>在某个范围之内(含最小、最大值)</u>
<u>in(...)</u>	<u>在in之后的列表中的值, 多选一</u>
<u>like 占位符</u>	<u>模糊匹配(匹配单个字符, %匹配任意个字符)</u>
<u>is null</u>	<u>是null</u>

常用的逻辑运算符如下：

逻辑运算符	功能
<u>and 或 &&</u>	<u>并且 (多个条件同时成立)</u>
<u>or 或 </u>	<u>或者 (多个条件任意一个成立)</u>
<u>not 或 !</u>	<u>非 , 不是</u>

案例1: 查询 姓名 为 杨逍 的员工

```
1  select id, username, password, name, gender, image, job, entrydate,
   create_time, update_time
2  from tb_emp
3  where name = '杨逍'; -- 字符串使用''或""包含
```

```
-- ===== DQL: 条件查询 =====
-- 1. 查询 姓名 为 杨逍 的员工
select id, username, password, name, gender, image, job, entrydate, create_time, update_time
from tb_emp
where name = '杨逍';
```

id	username	password	name	gender	image	job	entrydate	create_time	update_time
1	3 yangxiao	123456	杨逍		1 3.jpg	2	2008-05-01	2022-10-27 16:35:33	2022-10-27 16:35:33

案例2: 查询 id小于等于5 的员工信息

```
1 select id, username, password, name, gender, image, job, entrydate,
   create_time, update_time
2 from tb_emp
3 where id <=5;
```

```
-- 2. 查询 id小于等于5 的员工信息
select id, username, password, name, gender, image, job, entrydate, create_time, update_time
from tb_emp
where id <=5;
```

id	username	password	name	gender	image	job	entrydate	create_time	update_time
1	1 jinyong	123456	金庸		1 1.jpg	4	2000-01-01	2022-10-27 16:35:33	2022-10-27 16:35:33
2	2 zhangwuji	123456	张无忌		1 2.jpg	2	2015-01-01	2022-10-27 16:35:33	2022-10-27 16:35:33
3	3 yangxiao	123456	杨逍		1 3.jpg	2	2008-05-01	2022-10-27 16:35:33	2022-10-27 16:35:33
4	4 weiyixiao	123456	韦一笑		1 4.jpg	2	2007-01-01	2022-10-27 16:35:33	2022-10-27 16:35:33
5	5 changyuchun	123456	常遇春		1 5.jpg	2	2012-12-05	2022-10-27 16:35:33	2022-10-27 16:35:33

案例3: 查询 没有分配职位 的员工信息

```
1 select id, username, password, name, gender, image, job, entrydate,
   create_time, update_time
2 from tb_emp
3 where job is null ;
```

```
-- 3. 查询 没有分配职位 的员工信息
select id, username, password, name, gender, image, job, entrydate, create_time, update_time
from tb_emp
where job is null ;
```

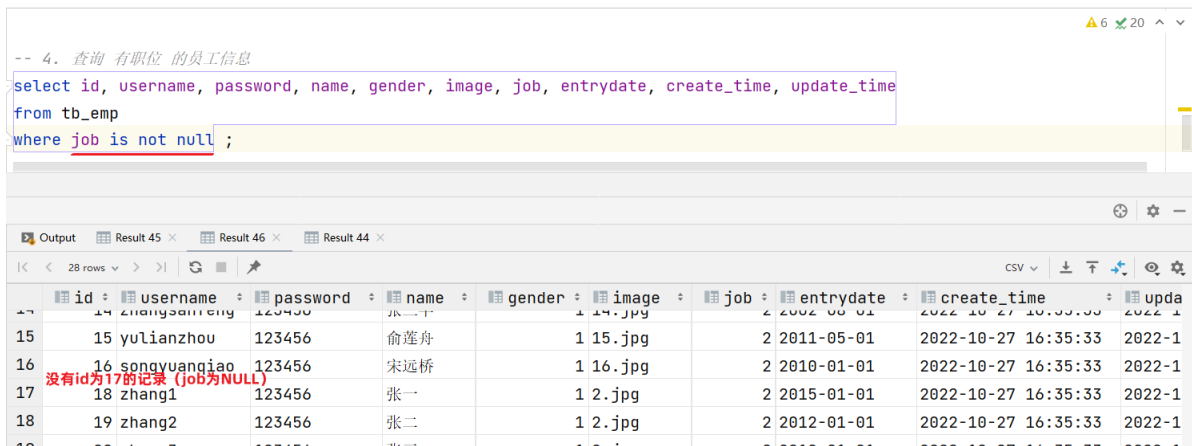
id	username	password	name	gender	image	job	entrydate	create_time	update_time
1	17 chenyouliang	12345678	陈友谅		1 17.jpg	<null>	2015-03-21	2022-10-27 16:35:33	2022-10-27 16:35:33

注意: 查询为NULL的数据时, 不能使用 `= null`



案例4: 查询 有职位 的员工信息

```
1  select id, username, password, name, gender, image, job, entrydate,
   create_time, update_time
2  from tb_emp
3  where job is not null ;
```



案例5: 查询 密码不等于 '123456' 的员工信息

```
1  -- 方式1:
2  select id, username, password, name, gender, image, job, entrydate,
   create_time, update_time
3  from tb_emp
4  where password <> '123456';
5  -- 方式2:
6  select id, username, password, name, gender, image, job, entrydate,
   create_time, update_time
7  from tb_emp
8  where password != '123456';
```

```
-- 5. 查询 密码不等于 '123456' 的员工信息
select id, username, password, name, gender, image, job, entrydate, create_time, update_time
from tb_emp
where password <> '123456';

select id, username, password, name, gender, image, job, entrydate, create_time, update_time
from tb_emp
where password != '123456';
```

id	username	password	name	gender	image	job	entrydate	create_time	update_time
1	17 chenyouliang	12345678	陈友谅		1 17.jpg	<null>	2015-03-21	2022-10-27 16:35:33	2022-10-

案例6: 查询 入职日期 在 '2000-01-01' (包含) 到 '2010-01-01' (包含) 之间的员工信息

```
1  -- 方式1:
2  select id, username, password, name, gender, image, job, entrydate,
3  create_time, update_time
4  from tb_emp
5  where entrydate>='2000-01-01' and entrydate<='2010-01-01';
6  -- 方式2: between...and
7  select id, username, password, name, gender, image, job, entrydate,
8  create_time, update_time
9  from tb_emp
10 where entrydate between '2000-01-01' and '2010-01-01';
```

```
-- 6. 查询 入职日期 在 '2000-01-01' (包含) 到 '2010-01-01' (包含) 之间的员工信息
select id, username, password, name, gender, image, job, entrydate, create_time, update_time
from tb_emp
where entrydate>='2000-01-01' and entrydate<='2010-01-01';

select id, username, password, name, gender, image, job, entrydate, create_time, update_time
from tb_emp
where entrydate between '2000-01-01' and '2010-01-01';
```

id	username	password	name	gender	image	job	entrydate	create_time	update_time
1	1 jinyong	123456	金庸		1 1.jpg	4	2000-01-01	2022-10-27 16:35:33	2022-1
2	3 yangxiao	123456	杨逍		1 3.jpg	2	2008-05-01	2022-10-27 16:35:33	2022-1
3	4 weiyixiao	123456	韦一笑		1 4.jpg	2	2007-01-01	2022-10-27 16:35:33	2022-1
4	7 jixiaofu	123456	纪晓芙		2 7.jpg	1	2005-08-01	2022-10-27 16:35:33	2022-1
5	11 luzhangke	123456	陆乘风		1 11.jpg	2	2007-02-01	2022-10-27 16:35:33	2022-1

案例7: 查询 入职时间 在 '2000-01-01' (包含) 到 '2010-01-01' (包含) 之间 且 性别为女的员工信息

```
1  select id, username, password, name, gender, image, job, entrydate,
2  create_time, update_time
3  from tb_emp
4  where entrydate between '2000-01-01' and '2010-01-01'
5  and gender = 2;
```

-- 7. 查询 入职时间在 '2000-01-01' (包含) 到 '2010-01-01' (包含) 之间 且 性别为女 的员工信息

```

select id, username, password, name, gender, image, job, entrydate, create_time, update_time
from tb_emp
where entrydate between '2000-01-01' and '2010-01-01' 条件1
      and gender = 2; 条件2

```

id	username	password	name	gender	image	job	entrydate	create_time	update_time
1	7 jixiaofu	123456	纪晓芙	2	7.jpg	1	2005-08-01	2022-10-27 16:35:33	2022-10-

案例8: 查询 职位是 2 (讲师), 3 (学工主管), 4 (教研主管) 的员工信息

```

1  -- 方式1: 使用or连接多个条件
2  select id, username, password, name, gender, image, job, entrydate,
3  create_time, update_time
4  from tb_emp
5  where job=2 or job=3 or job=4;
6  -- 方式2: in关键字
7  select id, username, password, name, gender, image, job, entrydate,
8  create_time, update_time
9  from tb_emp
10 where job in (2,3,4);

```

-- 8. 查询 职位是 2 (讲师), 3 (学工主管), 4 (教研主管) 的员工信息

```

select id, username, password, name, gender, image, job, entrydate, create_time, update_time
from tb_emp
where job=2 or job=3 or job=4;

select id, username, password, name, gender, image, job, entrydate, create_time, update_time
from tb_emp
where job in (2,3,4);

```

id	username	password	name	gender	image	job	entrydate	create_time	update_time
1	1 jinyong	123456	金庸	1	1.jpg	4	2000-01-01	2022-10-27 16:35:33	2022-1
2	2 zhangwuji	123456	张无忌	1	2.jpg	2	2015-01-01	2022-10-27 16:35:33	2022-1
3	3 yangxiao	123456	杨逍	1	3.jpg	2	2008-05-01	2022-10-27 16:35:33	2022-1
4	4 weivixian	123456	韦一笑	1	4.png	2	2007-01-01	2022-10-27 16:35:33	2022-1

案例9: 查询 姓名 为两个字的员工信息

```

1  select id, username, password, name, gender, image, job, entrydate,
2  create_time, update_time
3  from tb_emp
4  where name like '__'; # 通配符 " " 代表任意1个字符

```

```
-- 9. 查询 姓名 为两个字的员工信息
select id, username, password, name, gender, image, job, entrydate, create_time, update_time
from tb_emp
where name like '___'; 模糊查询中，通配符 "_" 表示1个任意字符
```

	id	username	password	name	gender	image	job	entrydate	create_time	update_time
1	1	jinyong	123456	金庸		1 1.jpg	4	2000-01-01	2022-10-27 16:35:33	2022-10-27 16:35:33
2	3	yangxiao	123456	杨逍		1 3.jpg	2	2008-05-01	2022-10-27 16:35:33	2022-10-27 16:35:33
3	6	xiaozhao	123456	小昭		2 6.jpg	3	2013-09-05	2022-10-27 16:35:33	2022-10-27 16:35:33
4	10	zhaomin	123456	张敏		2 10.jpg	1	2013-09-05	2022-10-27 16:35:33	2022-10-27 16:35:33

案例10：查询 姓 '张' 的员工信息

```
1 select id, username, password, name, gender, image, job, entrydate,
   create_time, update_time
2 from tb_emp
3 where name like '张%'; # 通配符 "%" 代表任意个字符（0个 ~ 多个）
```

```
-- 10. 查询 姓 '张' 的员工信息
select id, username, password, name, gender, image, job, entrydate, create_time, update_time
from tb_emp
where name like '张%'; # 通配符 "%" 代表任意个字符（0个 ~ 多个）
```

	id	username	password	name	gender	image	job	entrydate	create_time	update_time
1	2	zhangwuji	123456	张无忌		1 2.jpg	2	2015-01-01	2022-10-27 16:35:33	2022-10-27 16:35:33
2	14	zhangsanfeng	123456	张三丰		1 14.jpg	2	2002-08-01	2022-10-27 16:35:33	2022-10-27 16:35:33
3	18	zhang1	123456	张三		1 18.jpg	2	2015-01-01	2022-10-27 16:35:33	2022-10-27 16:35:33

1.5 聚合函数

之前我们做的查询都是横向查询，就是根据条件一行一行的进行判断，而使用聚合函数查询就是纵向查询，它是对一列的值进行计算，然后返回一个结果值。（将一列数据作为一个整体，进行纵向计算）

语法：

```
1 select 聚合函数(字段列表) from 表名 ;
```

注意：聚合函数会忽略空值，对NULL值不作为统计。

常用聚合函数：

函数	功能
<u>count</u>	<u>统计数量</u>
<u>max</u>	<u>最大值</u>
<u>min</u>	<u>最小值</u>
<u>avg</u>	<u>平均值</u>
<u>sum</u>	<u>求和</u>

count : 按照列去统计有多少行数据。

- 在根据指定的列统计的时候, 如果这一列中有null的行, 该行不会被统计在其中。

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

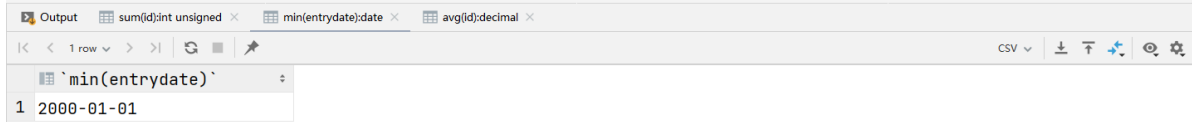
案例1: 统计该企业员工数量

```
1  # count(字段)
2  select count(id) from tb_emp;-- 结果: 29
3  select count(job) from tb_emp;-- 结果: 28 (聚合函数对NULL值不做计算)
4
5  # count(常量)
6  select count(0) from tb_emp;
7  select count('A') from tb_emp;
8
9  # count(*) 推荐此写法 (MySQL底层进行了优化)
10 select count(*) from tb_emp;
```

案例2: 统计该企业最早入职的员工

```
1  select min(entrydate) from tb_emp;
```

```
-- 2. 统计该企业最早入职的员工
select min(entrydate) from tb_emp;
```



	`min(entrydate)`
1	2000-01-01

案例3：统计该企业最迟入职的员工

```
1 select max(entrydate) from tb_emp;
```

```
-- 3. 统计该企业最迟入职的员工
select max(entrydate) from tb_emp;
```



	`max(entrydate)`
1	2020-01-01

案例4：统计该企业员工 ID 的平均值

```
1 select avg(id) from tb_emp;
```

```
-- 4. 统计该企业员工 ID 的平均值
select avg(id) from tb_emp;
```



	`avg(id)`
1	15.0000

案例5：统计该企业员工的 ID 之和

```
1 select sum(id) from tb_emp;
```

```
-- 5. 统计该企业员工的 ID 之和
select sum(id) from tb_emp;
```



	`sum(id)`
1	435

1.6 分组查询

分组：按照某一列或者某几列，把相同的数据进行合并输出。

分组其实就是按列进行分类(指定列下相同的数据归为一类)，然后可以对分类完的数据进行合并计算。

分组查询通常会使用聚合函数进行计算。

语法：

```
1  select 字段列表 from 表名 [where 条件] group by 分组字段名 [having  
   分组后过滤条件];
```

案例1：根据性别分组，统计男性和女性员工的数量

```
1  select gender, count(*)  
2  from tb_emp  
3  group by gender; -- 按照gender字段进行分组（gender字段下相同的数据归为一  
   组）
```

-- 1. 根据性别分组，统计男性和女性员工的数量

```
select gender, count(*)  
from tb_emp  
group by gender;
```

gender	count(*)
1	24
2	5

案例2：查询入职时间在 '2015-01-01'（包含）以前的员工，并对结果根据职位分组，获取员工数量大于等于2的职位

```
1  select job, count(*)  
2  from tb_emp  
3  where entrydate <= '2015-01-01' -- 分组前条件  
4  group by job -- 按照job字段分组  
5  having count(*) >= 2; -- 分组后条件
```

-- 3. 先查询入职时间在 '2015-01-01' (包含) 以前的员工, 并对结果根据职位分组, 获取员工数量大于等于2的职位

```

select job, count(*)
from tb_emp
where entrydate <= '2015-01-01' 分组前: 筛选出入职时间在'2015-01-01'之前的员工
group by job 按照职位分组 (job字段下相同的数据划分为一组)
having count(*) >= 2; 分组后: 筛选出各组中员工数量>=2的 (仅保留满足条件的组)

```

Output Result 72 x Result 73 x Result 71 x

2 rows

	job	count(*)
1	2	18
2	1	5

注意事项:

- 分组之后, 查询的字段一般为聚合函数和分组字段, 查询其他字段无任何意义
- 执行顺序: where > 聚合函数 > having

where与having区别 (面试题)

- 执行时机不同: where是分组之前进行过滤, 不满足where条件, 不参与分组; 而having是分组之后对结果进行过滤。
- 判断条件不同: where不能对聚合函数进行判断, 而having可以。

1.7 排序查询

排序在日常开发中是非常常见的一个操作, 有升序排序, 也有降序排序。

语法:

```

1  select  字段列表
2  from    表名
3  [where  条件列表]
4  [group by  分组字段 ]
5  order by  字段1  排序方式1 ,  字段2  排序方式2 ... ;

```

- 排序方式:
 - ASC : 升序 (默认值)
 - DESC: 降序

案例1: 根据入职时间, 对员工进行升序排序


```

1  select id, username, password, name, gender, image, job, entrydate,
   create_time, update_time
2  from tb_emp
3  order by entrydate ASC; -- 按照entrydate字段下的数据进行升序排序
4
5  select id, username, password, name, gender, image, job, entrydate,
   create_time, update_time
6  from tb_emp
7  order by entrydate; -- 默认就是ASC（升序）

```

-- 1. 根据入职时间，对员工进行升序排序

```

select id, username, password, name, gender, image, job, entrydate, create_time, update_time
from tb_emp
order by entrydate ASC;

select id, username, password, name, gender, image, job, entrydate, create_time, update_time
from tb_emp
order by entrydate; -- 默认就是ASC（升序）

```

	id	username	password	name	gender	image	job	entrydate	create_time
1	1	jinyong	123456	金庸	1	1.jpg	4	2000-01-01	2022-10-27 16:35:33
2	25	zhang8	123456	张八	1	2.jpg	2	2002-01-01	2022-10-27 16:35:33
3	14	zhangsanfeng	123456	张三丰	1	14.jpg	2	2002-08-01	2022-10-27 16:35:33
4	27	zhang10	123456	张十	1	2.jpg	2	2004-01-01	2022-10-27 16:35:33
5	7	jixiaofu	123456	纪晓芙	2	7.jpg	1	2005-08-01	2022-10-27 16:35:33
6	24	zhang7	123456	张七	1	2.jpg	2	2006-01-01	2022-10-27 16:35:33

注意事项：如果是升序，可以不指定排序方式ASC

案例2：根据入职时间，对员工进行降序排序

```

1  select id, username, password, name, gender, image, job, entrydate,
   create_time, update_time
2  from tb_emp
3  order by entrydate DESC; -- 按照entrydate字段下的数据进行降序排序

```

-- 2. 根据入职时间，对员工进行降序排序

```

select id, username, password, name, gender, image, job, entrydate, create_time, update_time
from tb_emp
order by entrydate DESC; -- 按照entrydate字段下的数据进行降序排序

```

	id	username	password	name	gender	image	job	entrydate	create_time
1	29	zhang12	123456	张十二	1	2.jpg	2	2020-01-01	2022-10-27 16:35:33
2	20	zhang3	123456	张三	1	2.jpg	2	2018-01-01	2022-10-27 16:35:33
3	22	zhang5	123456	张五	1	2.jpg	2	2016-01-01	2022-10-27 16:35:33
4	17	chenyouliang	12345678	陈友谅	1	17.jpg	<null>	2015-03-21	2022-10-27 16:35:33
5	2	zhangwudi	123456	张五弟	1	2.jpg	2	2015-01-01	2022-10-27 16:35:33

案例3：根据入职时间对公司的员工进行升序排序，入职时间相同，再按照更新时间进行降序排序

```

1  select id, username, password, name, gender, image, job, entrydate,
   create_time, update_time
2  from tb_emp
3  order by entrydate ASC , update_time DESC;

```

-- 3. 根据 入职时间 对公司的员工进行 升序排序，入职时间相同，再按照 更新时间 进行降序排序

```

select id, username, password, name, gender, image, job, entrydate, create_time, update_time
from tb_emp
order by entrydate ASC , update_time DESC;

```

	password	name	gender	image	job	entrydate	create_time	update_time
1	123456	金庸	1	1.jpg	4	2000-01-01	2022-10-27 16:35:33	2022-10-27 16:35:35
2	123456	张八	1	2.jpg	2	2002-01-01	2022-10-27 16:35:33	2022-10-27 16:36:23
3	123456	张三丰	1	14.jpg	2	2002-08-01	2022-10-27 16:35:33	2022-10-27 16:36:01
4	123456	张十	1	2.jpg	2	2004-01-01	2022-10-27 16:35:33	2022-10-27 16:36:27
5	123456	纪晓芙	2	7.jpg	1	2005-08-01	2022-10-27 16:35:33	2022-10-27 16:35:47
6	123456	张七	1	2.jpg	2	2006-01-01	2022-10-27 16:35:33	2022-10-27 16:36:21
7	123456	张十一	1	2.jpg	2	2007-01-01	2022-10-27 16:35:33	2022-10-27 16:36:29
8	123456	韦一笑	1	4.jpg	2	2007-01-01	2022-10-27 16:35:33	2022-10-27 16:35:41
9	123456	周芷若	1	11.jpg	2	2007-02-01	2022-10-27 16:35:33	2022-10-27 16:35:55

注意事项：如果是多字段排序，当第一个字段值相同时，才会根据第二个字段进行排序

1.8 分页查询

分页操作在业务系统开发时，也是非常常见的一个功能，日常我们在网站中看到的各种各样的分页条，后台也都需要借助于数据库的分页操作。

课程编号	课程学科	课程名称	价格(元)	适用人群	课程介绍	创建时间	操作
9qd138kq	Java	初级Java	10000	小白学员	初级Java从小白开始	2021-06-02 18:19:58	修改 删除
99vhgujw	Java	Java基础班	10001	中级程序员	基础班	2021-06-02 18:22:21	修改 删除
1bxepowu	Java	Java从小白到大神	10005	小白学员	从小白到大神的进阶之路	2021-06-02 18:24:45	修改 删除
sd2e9xow	UI设计	UI设计课程新课程	99999	小白学员	UI设计课程新课程	2021-06-03 17:29:24	修改 删除
vnezu9st	人工智能	人工智能6月	3999	中级程序员	人工智能6月	2021-06-03 17:38:42	修改 删除
3vssua42	UI设计	UI课程一	99999	小白学员	文强文强文强为请问问	2021-06-03 17:57:35	修改 删除
pddu2qcz	新媒体	新媒体测试	1888	小白学员	新媒体测试新媒体测试	2021-07-15 23:05:50	修改 删除
9h9oqsfg	UI设计	UI设计快人一步	88888	小白学员	小白快速上手完成独立完成任务	2021-11-08 22:28:21	修改 删除
96u9u5mv	前端	高级前端课程	1888	中级程序员	高级前端课程	2021-11-10 13:56:44	修改 删除
p7m47cko	Java	Java高级课程	9999	中级程序员	Java高级课程	2021-11-15 14:58:37	修改 删除

共 20 条 10条/页 < 1 2 > 前往 2 页

分页查询语法：

```

1  select 字段列表 from 表名 limit 起始索引, 查询记录数 ;

```

案例1：从起始索引0开始查询员工数据，每页展示5条记录

```
1 select id, username, password, name, gender, image, job, entrydate,
   create_time, update_time
2 from tb_emp
3 limit 0 , 5; -- 从索引0开始，向后取5条记录
```

-- 1. 从起始索引0开始查询员工数据，每页展示5条记录

```
select id, username, password, name, gender, image, job, entrydate, create_time, update_time
from tb_emp
limit 0 , 5;
```

	id	username	password	name	gender	image	job	entrydate	create_time	update_time
1	1	jinyong	123456	金庸	1	1.jpg	4	2000-01-01	2022-10-27 16:35:33	2022-10-27 16:35:33
2	2	zhangwuji	123456	张无忌	1	2.jpg	2	2015-01-01	2022-10-27 16:35:33	2022-10-27 16:35:33
3	3	yangxiao	123456	杨逍	1	3.jpg	2	2008-05-01	2022-10-27 16:35:33	2022-10-27 16:35:33
4	4	weiyixiao	123456	韦一笑	1	4.jpg	2	2007-01-01	2022-10-27 16:35:33	2022-10-27 16:35:33
5	5	changyuchun	123456	常遇春	1	5.jpg	2	2012-12-05	2022-10-27 16:35:33	2022-10-27 16:35:33

案例2：查询 第1页 员工数据，每页展示5条记录

```
1 select id, username, password, name, gender, image, job, entrydate,
   create_time, update_time
2 from tb_emp
3 limit 5; -- 如果查询的是第1页数据，起始索引可以省略，直接简写为：limit 条数
```

```
select id, username, password, name, gender, image, job, entrydate, create_time, update_time
from tb_emp
limit 5; -- 如果查询的是第1页数据，起始索引可以省略，直接简写为：limit 条数
```

	id	username	password	name	gender	image	job	entrydate	create_time	update_time
1	1	jinyong	123456	金庸	1	1.jpg	4	2000-01-01	2022-10-27 16:35:33	2022-10-27 16:35:33
2	2	zhangwuji	123456	张无忌	1	2.jpg	2	2015-01-01	2022-10-27 16:35:33	2022-10-27 16:35:33
3	3	yangxiao	123456	杨逍	1	3.jpg	2	2008-05-01	2022-10-27 16:35:33	2022-10-27 16:35:33
4	4	weiyixiao	123456	韦一笑	1	4.jpg	2	2007-01-01	2022-10-27 16:35:33	2022-10-27 16:35:33
5	5	changyuchun	123456	常遇春	1	5.jpg	2	2012-12-05	2022-10-27 16:35:33	2022-10-27 16:35:33

案例3：查询 第2页 员工数据，每页展示5条记录

```
1 select id, username, password, name, gender, image, job, entrydate,
   create_time, update_time
2 from tb_emp
3 limit 5 , 5; -- 从索引5开始，向后取5条记录
```

```

select id, username, password, name, gender, image, job, entrydate, create_time, update_time
from tb_emp
limit 5, 5;

```

	id	username	password	name	gender	image	job	entrydate	create_time	update_time
1	6	xiaozhao	123456	小昭	2	6.jpg	3	2013-09-05	2022-10-27 16:35:33	2022-10-27 16:35:33
2	7	jixiaofu	123456	纪晓芙	2	7.jpg	1	2005-08-01	2022-10-27 16:35:33	2022-10-27 16:35:33
3	8	zhouzhiyue	123456	周芷若	2	8.jpg	1	2014-11-09	2022-10-27 16:35:33	2022-10-27 16:35:33
4	9	dingminjun	123456	丁敏君	2	9.jpg	1	2011-03-11	2022-10-27 16:35:33	2022-10-27 16:35:33
5	10	zhaomin	123456	赵敏	2	10.jpg	1	2013-09-05	2022-10-27 16:35:33	2022-10-27 16:35:33

案例4：查询 第3页 员工数据， 每页展示5条记录

```

1  select id, username, password, name, gender, image, job, entrydate,
   create_time, update_time
2  from tb_emp
3  limit 10, 5; -- 从索引10开始， 向后取5条记录

```

```

select id, username, password, name, gender, image, job, entrydate, create_time, update_time
from tb_emp
limit 10, 5;

```

	id	username	password	name	gender	image	job	entrydate	create_time	update_time
1	11	luzhangke	123456	鹿杖客	1	11.jpg	2	2007-02-01	2022-10-27 16:35:33	2022-10-27 16:35:33
2	12	hebiweng	123456	鹤笔翁	1	12.jpg	2	2008-08-18	2022-10-27 16:35:33	2022-10-27 16:35:33
3	13	fangdongbai	123456	方东白	1	13.jpg	1	2012-11-01	2022-10-27 16:35:33	2022-10-27 16:35:33
4	14	zhangsanfeng	123456	张三丰	1	14.jpg	2	2002-08-01	2022-10-27 16:35:33	2022-10-27 16:35:33
5	15	yulianzhou	123456	俞莲舟	1	15.jpg	2	2011-05-01	2022-10-27 16:35:33	2022-10-27 16:35:33

注意事项：

1. 起始索引从0开始。 计算公式： 起始索引 = (查询页码 - 1) * 每页显示记录数
2. 分页查询是数据库的方言，不同的数据库有不同的实现，MySQL中是LIMIT
3. 如果查询的是第一页数据，起始索引可以省略，直接简写为 limit 条数

1.9 案例

DQL的基本语法我们学习结束了，接下来我们就运用所掌握的DQL语句的语法来完成两个案例。

1.9.1 案例一

案例：根据需求完成员工管理的条件分页查询

姓名张

性别男

入职时间 从 到

查询

+ 新增员工

- 批量删除

<input type="checkbox"/>	姓名	用户名	性别	职位	入职日期	最后操作时间	操作
<input type="checkbox"/>	赵敬	zhaomin	女	班主任	2008-12-18	2022-07-22 12:05:20	编辑 删除
<input type="checkbox"/>	风清扬	fengqingyang	男	讲师	2015-07-22	2022-07-21 15:00:00	编辑 删除
<input type="checkbox"/>	风清扬	fengqingyang	男	讲师	2015-07-22	2022-07-21 15:00:00	编辑 删除
<input type="checkbox"/>	风清扬	fengqingyang	男	讲师	2015-07-22	2022-07-21 15:00:00	编辑 删除
<input type="checkbox"/>	风清扬	fengqingyang	男	讲师	2015-07-22	2022-07-21 15:00:00	编辑 删除
<input type="checkbox"/>	风清扬	fengqingyang	男	讲师	2015-07-22	2022-07-21 15:00:00	编辑 删除
<input type="checkbox"/>	风清扬	fengqingyang	男	讲师	2015-07-22	2022-07-21 15:00:00	编辑 删除
<input type="checkbox"/>	风清扬	fengqingyang	男	讲师	2015-07-22	2022-07-21 15:00:00	编辑 删除
<input type="checkbox"/>	风清扬	fengqingyang	男	讲师	2015-07-22	2022-07-21 15:00:00	编辑 删除
<input type="checkbox"/>	风清扬	fengqingyang	男	讲师	2015-07-22	2022-07-21 15:00:00	编辑 删除

每页展示记录数

共500条数据

<

1

2

3

4

5

...

50

>

跳至 页

分析：根据输入的条件，查询第1页数据

- 在员工管理的列表上方有一些查询条件：员工姓名、员工性别，员工入职时间（开始时间~结束时间）
 - 姓名：张
 - 性别：男
 - 入职时间：2000-01-01 ~ 2015-12-31
- 除了查询条件外，在列表的下面还有一个分页条，这就涉及到了分页查询
 - 查询第1页数据（每页显示10条数据）
- 基于查询的结果，按照修改时间进行降序排序

结论：条件查询 + 分页查询 + 排序查询

SQL语句代码：

```

1  -- 根据输入条件查询第1页数据（每页展示10条记录）
2  -- 输入条件：
3      -- 姓名：张 （模糊查询）
4      -- 性别：男
5      -- 入职时间：2000-01-01 ~ 2015-12-31
6  -- 分页： 0 , 10
7  -- 排序： 修改时间  DESC
8  select id, username, password, name, gender, image, job, entrydate,
9      create_time, update_time
10 from tb_emp
11 where name like '张%' and gender = 1 and entrydate between '2000-01-
12     01' and '2015-12-31'
13 order by update_time desc
14 limit 0 , 10;

```

```
select id, username, password, name, gender, image, job, entrydate, create_time, update_time
from tb_emp
where name like '张%' and gender = 1 and entrydate between '2000-01-01' and '2015-12-31'
order by update_time desc
limit 0 , 10;
```

OutputResult 87Result 881. 从起始索引0开始查询员工数据, 每页展示5条记录

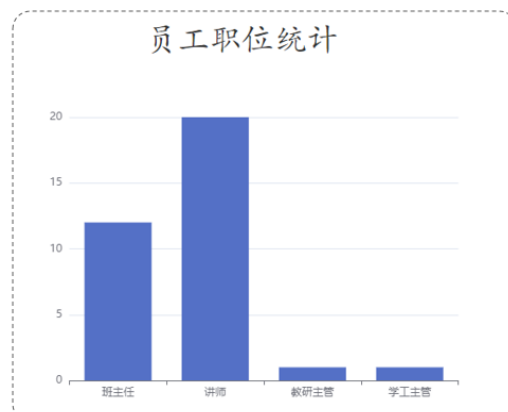
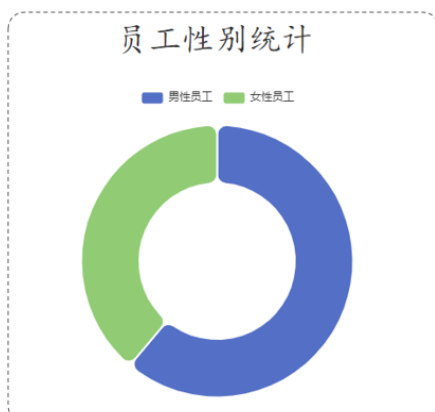
10 rows

CSVDownloadCopy

	id	username	password	name	gender	image	job	entrydate	create_time	update_time
1	28	zhang11	123456	张十一		1 2.jpg	2	2007-01-01	2022-10-27 16:35:33	2022-10-27 16:36:29
2	27	zhang10	123456	张十		1 2.jpg	2	2004-01-01	2022-10-27 16:35:33	2022-10-27 16:36:27
3	26	zhang9	123456	张九		1 2.jpg	2	2011-01-01	2022-10-27 16:35:33	2022-10-27 16:36:25
4	25	zhang8	123456	张八		1 2.jpg	2	2002-01-01	2022-10-27 16:35:33	2022-10-27 16:36:23
5	24	zhang7	123456	张七		1 2.jpg	2	2006-01-01	2022-10-27 16:35:33	2022-10-27 16:36:21
6	23	zhang6	123456	张六		1 2.jpg	2	2012-01-01	2022-10-27 16:35:33	2022-10-27 16:36:19
7	21	zhang4	123456	张四		1 2.jpg	2	2015-01-01	2022-10-27 16:35:33	2022-10-27 16:36:15
8	19	zhang2	123456	张二		1 2.jpg	2	2012-01-01	2022-10-27 16:35:33	2022-10-27 16:36:11
9	18	zhang1	123456	张一		1 2.jpg	2	2015-01-01	2022-10-27 16:35:33	2022-10-27 16:36:09
10	14	zhangsanfeng	123456	张三丰		1 14.jpg	2	2002-08-01	2022-10-27 16:35:33	2022-10-27 16:36:01

1.9.2 案例二

案例：根据需求完成员工信息的统计



分析：以上信息统计在开发中也叫图形报表（将统计好的数据以可视化的形式展示出来）

- 员工性别统计：以饼状图的形式展示出企业男性员工人数和女性员工人数
 - 只要查询出男性员工和女性员工各自有多少人就可以了
- 员工职位统计：以柱状图的形式展示各职位的在岗人数
 - 只要查询出各个职位有多少人就可以了

员工性别统计：

```
1  -- if(条件表达式, true取值 , false取值)
2  select if(gender=1,'男性员工','女性员工') AS 性别, count(*) AS 人数
3  from tb_emp
4  group by gender;
```

-- 案例1：根据需求完成员工性别信息的统计

```
-- if(条件表达式, true取值 , false取值)
select if(gender=1,'男性员工','女性员工') AS 性别, count(*) AS 人数
from tb_emp
group by gender;
```

Output Result 87 × Result 88 × Result 89 ×

2 rows

	性别	人数
1	男性员工	24
2	女性员工	5

if(表达式, tvalue, fvalue)：当表达式为true时，取值tvalue；当表达式为false时，取值fvalue

员工职位统计：

```

1  -- case 表达式 when 值1 then 结果1  when 值2  then 结果2 ...  else
   result  end
2  select (case job
3           when 1 then '班主任'
4           when 2 then '讲师'
5           when 3 then '学工主管'
6           when 4 then '教研主管'
7           else '未分配职位'
8         end) AS 职位 ,
9         count(*) AS 人数
10 from tb_emp
11 group by job;

```

```

select (case job
        when 1 then '班主任'
        when 2 then '讲师'
        when 3 then '学工主管'
        when 4 then '教研主管'
        else '未分配职位'
      end) AS 职位 ,
      count(*) AS 人数
from tb_emp
group by job;

```

职位	人数
1 教研主管	1
2 讲师	21
3 学工主管	1
4 班主任	5
5 未分配职位	1

```

case  表达式      when  值1    then  结果1    [when 值2  then  结果2 ...]
[else result]      end

```

2. 多表设计

关于单表的操作(单表的设计、单表的增删改查)我们就已经学习完了。接下来我们就要来学习多表的操作，首先来学习多表的设计。

项目开发中，在进行数据库表结构设计时，会根据业务需求及业务模块之间的关系，分析并设计表结构，由于业务之间相互关联，所以各个表结构之间也存在着各种联系，基本上分为三种：

- 一对多 (多对一)
- 多对多
- 一对一

2.1 一对多

2.1.1 表设计

需求：根据页面原型及需求文档，完成部门及员工的表结构设计

- 员工管理页面原型：（前面已完成tb_emp表结构设计）

员工管理

姓名

请输入员工姓名

性别

请选择

入职时间

从

开始时间

至

结束时间

查询

+ 新增员工

- 批量删除

<input type="checkbox"/>	姓名	图像	性别	职位	入职日期	最后操作时间	操作
<input type="checkbox"/>	赵敏		女	班主任	2008-12-18	2022-07-22 12:05:20	编辑 删除
<input type="checkbox"/>	风清扬		男	讲师	2015-07-22	2022-07-21 15:00:00	编辑 删除
<input type="checkbox"/>	风清扬		男	讲师	2015-07-22	2022-07-21 15:00:00	编辑 删除
<input type="checkbox"/>	风清扬		男	讲师	2015-07-22	2022-07-21 15:00:00	编辑 删除
<input type="checkbox"/>	风清扬		男	讲师	2015-07-22	2022-07-21 15:00:00	编辑 删除
<input type="checkbox"/>	风清扬		男	讲师	2015-07-22	2022-07-21 15:00:00	编辑 删除
<input type="checkbox"/>	风清扬		男	讲师	2015-07-22	2022-07-21 15:00:00	编辑 删除
<input type="checkbox"/>	风清扬		男	讲师	2015-07-22	2022-07-21 15:00:00	编辑 删除
<input type="checkbox"/>	风清扬		男	讲师	2015-07-22	2022-07-21 15:00:00	编辑 删除
<input type="checkbox"/>	风清扬		男	讲师	2015-07-22	2022-07-21 15:00:00	编辑 删除

每页展示记录数

10

共500条数据

<

1

2

3

4

5

...

50

>

跳至

1

页

- 部门管理页面原型:

部门管理

[illegible]

页面开发规则

1. 部门查询

1.1 查询全部数据（由于部门数据比较少，不考虑分页）。

2. 新增部门

1.1 点击新增部门，会打开新增部门的页面。

1.2 部门名称，必填，唯一，长度为2-10位。

3. 删除部门

弹出确认框，提示“您确定要删除该部门的信息吗？”如果选择确定，则删除该部门，删除成功后，重新刷新列表页面。 如果选择了 取消， 则不执行任何操作。

经过上述分析，现已明确的部门表结构：

- 业务字段： 部门名称
- 基础字段： id(主键)、创建时间、修改时间

部门表 - SQL语句：

```
1  # 建议：创建新的数据库（多表设计存放在新数据库下）
2  create database db03;
3  use db03;
4
5  -- 部门表
6  create table tb_dept
7  (
8      id int unsigned primary key auto_increment comment '主键ID',
9      name varchar(10) not null unique comment '部门名称',
10     create_time datetime not null comment '创建时间',
11     update_time datetime not null comment '修改时间'
12 ) comment '部门表';
```

部门表创建好之后，我们还需要再修改下员工表。为什么要修改员工表呢？是因为我们之前设计员工表（单表）的时候，并没有考虑员工的归属部门。

新增员工

* 用户名

请输入用户名, 2-20字符, 不可重复

* 员工姓名

请输入员工姓名, 2-10个字

* 性 别

请选择

图 像

职 位

请选择

入职日期

2022-07-22

归属部门

请选择 员工需要关联部门

保存

取消

修改员工

* 用户名

zhaomin

* 员工姓名

赵敏

* 性 别

女

图 像

职 位

班主任

入职日期

2008-12-18

归属部门

学工部

保存

取消

员工表：添加归属部门字段

```
1  -- 员工表
2  create table tb_emp
3  (
4      id          int unsigned primary key auto_increment comment
        'ID',
5      username    varchar(20)          not null unique comment '用户名',
6      password    varchar(32) default '123456' comment '密码',
7      name        varchar(10)          not null comment '姓名',
8      gender      tinyint unsigned not null comment '性别，说明：1 男，
        2 女',
9      image       varchar(300) comment '图像',
10     job         tinyint unsigned comment '职位，说明：1 班主任,2 讲师,
        3 学工主管, 4 教研主管',
11     entrydate   date comment '入职时间',
12
13     dept_id     int unsigned comment '部门ID', -- 员工的归属部门
14
15     create_time datetime          not null comment '创建时间',
16     update_time datetime          not null comment '修改时间'
17 ) comment '员工表';
```

测试数据：

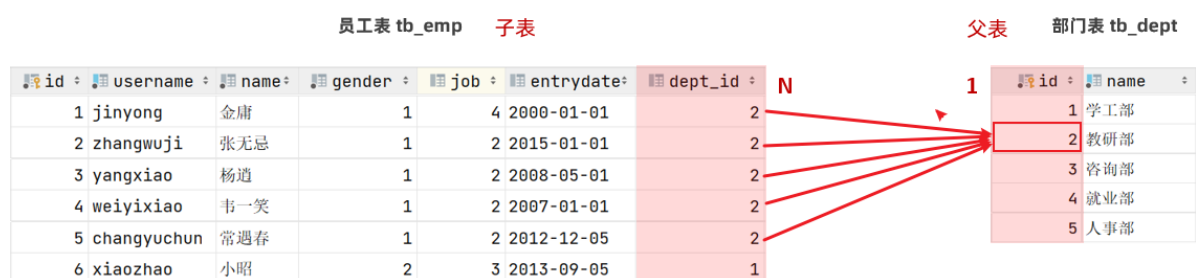
```
1  -- 部门表测试数据
2  insert into tb_dept (id, name, create_time, update_time) values
3  (1, '学工部', now(), now()),
4  (2, '教研部', now(), now()),
5  (3, '咨询部', now(), now()),
6  (4, '就业部', now(), now()),
7  (5, '人事部', now(), now());
8
9  -- 员工表测试数据
10 INSERT INTO tb_emp
11 (id, username, password, name, gender, image, job,
    entrydate, dept_id, create_time, update_time) VALUES
12 (1, 'jinyong', '123456', '金庸', 1, '1.jpg', 4, '2000-01-01', 2, now(), now()),
13 (2, 'zhangwuji', '123456', '张无忌', 1, '2.jpg', 2, '2015-01-
    01', 2, now(), now()),
14 (3, 'yangxiao', '123456', '杨逍', 1, '3.jpg', 2, '2008-05-
    01', 2, now(), now()),
15 (4, 'weiyixiao', '123456', '韦一笑', 1, '4.jpg', 2, '2007-01-
    01', 2, now(), now()),
```

```

16 (5, 'changyuchun', '123456', '常遇春', 1, '5.jpg', 2, '2012-12-
    05', 2, now(), now()),
17 (6, 'xiaozhao', '123456', '小昭', 2, '6.jpg', 3, '2013-09-
    05', 1, now(), now()),
18 (7, 'jixiaofu', '123456', '纪晓芙', 2, '7.jpg', 1, '2005-08-
    01', 1, now(), now()),
19 (8, 'zhouzhiruo', '123456', '周芷若', 2, '8.jpg', 1, '2014-11-
    09', 1, now(), now()),
20 (9, 'dingminjun', '123456', '丁敏君', 2, '9.jpg', 1, '2011-03-
    11', 1, now(), now()),
21 (10, 'zhaomin', '123456', '赵敏', 2, '10.jpg', 1, '2013-09-
    05', 1, now(), now()),
22 (11, 'luzhangke', '123456', '鹿杖客', 1, '11.jpg', 1, '2007-02-
    01', 1, now(), now()),
23 (12, 'hebiweng', '123456', '鹤笔翁', 1, '12.jpg', 1, '2008-08-
    18', 1, now(), now()),
24 (13, 'fangdongbai', '123456', '方东白', 1, '13.jpg', 2, '2012-11-
    01', 2, now(), now()),
25 (14, 'zhangsanfeng', '123456', '张三丰', 1, '14.jpg', 2, '2002-08-
    01', 2, now(), now()),
26 (15, 'yulianzhou', '123456', '俞莲舟', 1, '15.jpg', 2, '2011-05-
    01', 2, now(), now()),
27 (16, 'songyuanqiao', '123456', '宋远桥', 1, '16.jpg', 2, '2010-01-
    01', 2, now(), now()),
28 (17, 'chenyouliang', '123456', '陈友谅', 1, '17.jpg', NULL, '2015-03-
    21', NULL, now(), now());

```

员工表 - 部门表之间的关系：



一对多关系实现：在数据库表中多的一方，添加字段，来关联属于一这方的主键。

2.1.2 外键约束

问题

- 表结构创建完毕后，我们看到两张表的数据分别为：

id	username	password	name	gender	image	job	entrydate	dept_id
1	jinyong	123456	金庸		1 1.jpg	4	2000-01-01	2
2	zhangwuji	123456	张无忌		1 2.jpg	2	2015-01-01	2
3	yangxiao	123456	杨逍		1 3.jpg	2	2008-05-01	2
4	weiyixiao	123456	韦一笑		1 4.jpg	2	2007-01-01	2
5	changyuchun	123456	常遇春		1 5.jpg	2	2012-12-05	2
6	xiaozhao	123456	小昭		2 6.jpg	3	2013-09-05	1
7	jixiaofu	123456	纪晓芙		2 7.jpg	1	2005-08-01	1
8	zhouzhiruo	123456	周芷若		2 8.jpg	1	2014-11-09	1
9	dingminjun	123456	丁敏君		2 9.jpg	1	2011-03-11	1
10	zhaomin	123456	赵敏		2 10.jpg	1	2013-09-05	1
11	luzhangke	123456	鹿杖客		1 11.jpg	5	2007-02-01	3
12	hebiweng	123456	鹤笔翁		1 12.jpg	5	2008-08-18	3
13	fangdongbai	123456	方东白		1 13.jpg	5	2012-11-01	3
14	zhangsanfeng	123456	张三丰		1 14.jpg	2	2002-08-01	2
15	yuLianzhou	123456	俞莲舟		1 15.jpg	2	2011-05-01	2
16	songyuanqiao	123456	宋远桥		1 16.jpg	2	2010-01-01	2
17	chenyouliang	123456	陈友谅		1 17.jpg	<null>	2015-03-21	<null>

id	name	create_time	update_time
1	学工部	2022-08-31 20:08:34	2022-08-31 20:08:34
2	教研部	2022-08-31 20:08:34	2022-08-31 20:08:34
3	咨询部	2022-08-31 20:08:34	2022-08-31 20:08:34
4	就业部	2022-08-31 20:08:34	2022-08-31 20:08:34
5	人事部	2022-08-31 20:08:34	2022-08-31 20:08:34

现在员工表中有五个员工都归属于1号部门(学工部)，当删除了1号部门后，数据变为：

id	username	password	name	gender	image	job	entrydate	dept_id
1	jinyong	123456	金庸		1 1.jpg	4	2000-01-01	2
2	zhangwuji	123456	张无忌		1 2.jpg	2	2015-01-01	2
3	yangxiao	123456	杨逍		1 3.jpg	2	2008-05-01	2
4	weiyixiao	123456	韦一笑		1 4.jpg	2	2007-01-01	2
5	changyuchun	123456	常遇春		1 5.jpg	2	2012-12-05	2
6	xiaozhao	123456	小昭		2 6.jpg	3	2013-09-05	1
7	jixiaofu	123456	纪晓芙		2 7.jpg	1	2005-08-01	1
8	zhouzhiruo	123456	周芷若		2 8.jpg	1	2014-11-09	1
9	dingminjun	123456	丁敏君		2 9.jpg	1	2011-03-11	1
10	zhaomin	123456	赵敏		2 10.jpg	1	2013-09-05	1
11	luzhangke	123456	鹿杖客		1 11.jpg	5	2007-02-01	3
12	hebiweng	123456	鹤笔翁		1 12.jpg	5	2008-08-18	3
13	fangdongbai	123456	方东白		1 13.jpg	5	2012-11-01	3
14	zhangsanfeng	123456	张三丰		1 14.jpg	2	2002-08-01	2
15	yuLianzhou	123456	俞莲舟		1 15.jpg	2	2011-05-01	2
16	songyuanqiao	123456	宋远桥		1 16.jpg	2	2010-01-01	2
17	chenyouliang	123456	陈友谅		1 17.jpg	<null>	2015-03-21	<null>

id	name
2	教研部
3	咨询部
4	就业部
5	人事部

1号部门被删除了，但是依然还有5个员工是属于1号部门的。此时：就出现数据的不完整、不一致了。

问题分析

目前上述的两张表(员工表、部门表)，在数据库层面，并未建立关联，所以是无法保证数据的一致性和完整性的

问题解决

想解决上述的问题呢，我们就可以通过数据库中的 外键约束 来解决。

外键约束：让两张表的数据建立连接，保证数据的一致性和完整性。

对应的关键字：foreign key

外键约束的语法：

```

1  -- 创建表时指定
2  create table 表名 (
3      字段名      数据类型,
4      ...
5      [constraint]    [外键名称] foreign key (外键字段名) references
        主表 (主表列名)
6  );
7
8
9  -- 建完表后，添加外键
10 alter table 表名 add constraint 外键名称 foreign key (外键字段名)
    references 主表 (主表列名);

```

那接下来，我们就为员工表的dept_id 建立外键约束，来关联部门表的主键。

方式1：通过SQL语句操作

```

1  -- 修改表： 添加外键约束
2  alter table tb_emp
3  add constraint fk_dept_id foreign key (dept_id) references
    tb_dept(id);

```

方式2：图形化界面操作

Modify Table

Table:

tb_emp

Comment:

员工表

Columns (11)

Keys (2)

Indexes (2)

Foreign Keys (1)

Grants

Name:

tb_emp_fk_dept_id 外键约束名字

Target table:

tb_dept 目标表 (主表)

Update rule:

no action

Delete rule:

no action

Columns:

From:

dept_id 外键字段

To:

id 主表主键

SQL Script

Action: Execute in database

Execute

Preview

Cancel

当我们添加外键约束时，我们得保证当前数据库表中的数据是完整的。 所以，我们需要将之前删除掉的数据再添加回来。

当我们添加了外键之后，再删除ID为1的部门，就会发现，此时数据库报错了，不允许删除。

	id	name	create_time	update_time
1	1	学工部	2022-08-31 20:08:34	2022-08-31 20:08:34
2	2	教研部	2022-08-31 20:08:34	2022-08-31 20:08:34
3	3	咨询部	2022-08-31 20:08:34	2022-08-31 20:08:34
4	4	就业部	2022-08-31 20:08:34	2022-08-31 20:08:34
5	5	人事部	2022-08-31 20:08:34	2022-08-31 20:08:34

[23000][1451] Cannot delete or update a parent row: a foreign key constraint fails ('db02`.`emp`, CONSTRAINT `fk_dept_id` FOREIGN KEY (`dept_id`) REFERENCES `dept` (`id`))

外键约束 (foreign key) : 保证了数据的完整性和一致性。

- 物理外键
 - 概念：使用foreign key定义外键关联另外一张表。
 - 缺点：
 - 影响增、删、改的效率（需要检查外键关系）。
 - 仅用于单节点数据库，不适用与分布式、集群场景。
 - 容易引发数据库的死锁问题，消耗性能。
- 逻辑外键
 - 概念：在业务层逻辑中，解决外键关联。
 - 通过逻辑外键，就可以很方便的解决上述问题。

在现在的企业开发中，很少会使用物理外键，都是使用逻辑外键。 甚至在一些数据库开发规范中，会明确指出禁止使用物理外键 foreign key

2.2 一对一

一对一关系表在实际开发中应用起来比较简单，通常是用来做单表的拆分，也就是将一张大表拆分成两张小表，将大表中的一些基础字段放在一张表当中，将其他的字段放在另外一张表当中，以此来提高数据的操作效率。

一对一的应用场景： 用户表 (基本信息+身份信息)

id	name	gender	phone	degree	nationality	birthday	idcard	issued	expire_begin	expire_end
1	白眉鹰王	1	18812340001	初中	汉	1960-11-06	100000100000100001	朝阳区公安局	2000-06-10	<null>
2	青翼蝠王	1	18812340002	大专	汉	1971-11-06	100000100000100002	静安区公安局	2005-06-10	2025-06-10
3	金毛狮王	1	18812340003	初中	汉	1963-11-06	100000100000100003	昌平区公安局	2006-06-10	<null>
4	紫衫龙王	2	18812340004	硕士	回	1980-11-06	100000100000100004	海淀区公安局	2008-06-10	2028-06-10

- 基本信息：用户的ID、姓名、性别、手机号、学历
- 身份信息：民族、生日、身份证号、身份证签发机关，身份证的有效期（开始时间、结束时间）

如果在业务系统当中，对用户的基本信息查询频率特别的高，但是对于用户的身份信息查询频率很低，此时出于提高查询效率的考虑，我就可以将这张大表拆分成两张小表，第一张表存放的是用户的基本信息，而第二张表存放的就是用户的身份信息。他们两者之间一对一的关系，一个用户只能对应一个身份证，而一个身份证也只能关联一个用户。

那么在数据库层面怎么去体现上述两者之间是一一对一的关系呢？

其实一对一我们可以看成一种特殊的一对多。一对多我们是怎么设计表关系的？是不是在多的一方添加外键。同样我们也可以通过外键来体现一对一之间的关系，我们只需要在任意一方来添加一个外键就可以了。

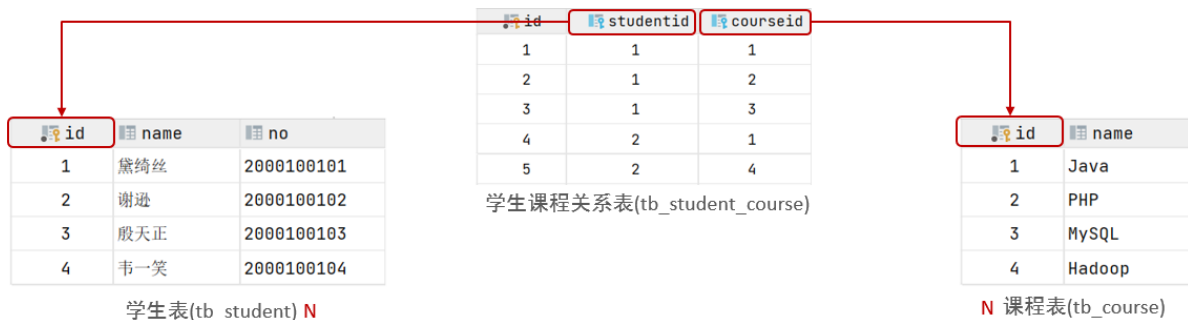

```
31         (4, '回', '1980-11-06', '100000100000100004', '海淀区公安局', '2008-06-10', '2028-06-10', 4);
```

2.3 多对多

多对多的关系在开发中属于也比较常见的。比如：学生和老师的关系，一个学生可以有多个授课老师，一个授课老师也可以有多个学生。在比如：学生和课程的关系，一个学生可以选修多门课程，一个课程也可以供多个学生选修。

案例：学生与课程的关系

- 关系：一个学生可以选修多门课程，一门课程也可以供多个学生选择
- 实现关系：建立第三张中间表，中间表至少包含两个外键，分别关联两方主键



SQL脚本：

```
1  -- 学生表
2  create table tb_student(
3      id int auto_increment primary key comment '主键ID',
4      name varchar(10) comment '姓名',
5      no varchar(10) comment '学号'
6  ) comment '学生表';
7  -- 学生表测试数据
8  insert into tb_student(name, no) values ('黛绮丝', '2000100101'), ('谢
9      逊', '2000100102'), ('殷天正', '2000100103'), ('韦一笑', '2000100104');
10
11 -- 课程表
12 create table tb_course(
13     id int auto_increment primary key comment '主键ID',
14     name varchar(10) comment '课程名称'
15 ) comment '课程表';
16 -- 课程表测试数据
17 insert into tb_course (name) values ('Java'), ('PHP'), ('MySQL') ,
18     ('Hadoop');
```

```

17
18  -- 学生课程表（中间表）
19  create table tb_student_course(
20      id int auto_increment comment '主键' primary key,
21      student_id int not null comment '学生ID',
22      course_id int not null comment '课程ID',
23      constraint fk_courseid foreign key (course_id) references
        tb_course (id),
24      constraint fk_studentid foreign key (student_id) references
        tb_student (id)
25  ) comment '学生课程中间表';
26  -- 学生课程表测试数据
27  insert into tb_student_course(student_id, course_id) values (1,1),
        (1,2), (1,3), (2,2), (2,3), (3,4);

```

2.4 案例

下面通过一个综合案例加深对于多表关系的理解，并掌握多表设计的流程。

需求

- 根据参考资料中提供的《**苍穹外卖_管理后台**》页面原型，设计分类管理、菜品管理、套餐管理模块的表结构。

步骤

1. 阅读页面原型及需求文档，分析各个模块涉及到的表结构，及表结构之间的关系。
2. 根据页面原型及需求文档，分析各个表结构中具体的字段及约束。

分析

- 页面原型-分类管理

分类管理

1 分类名称

请输入分类名称

分类类型

请选择分类类型

搜索

2

分类名称	分类类型	排序	状态	操作时间	3 操作
荤菜	菜品分类	1	启用	2019-01-01 11:11	修改 删除 禁用
素菜	菜品分类	2	禁用	2019-01-01 11:11	修改 删除 启用
凉菜	菜品分类	3	启用	2019-01-01 11:11	修改 删除 禁用
周一套餐	套餐分类	4	禁用	2019-01-02 11:11	修改 删除 启用

新增菜品分类

新增套餐分类

分类的信息：分类名称、分类类型[菜品/套餐]、分类排序、分类状态[禁用/启用]、分类的操作时间(修改时间)。

• 页面原型-菜品管理

菜品管理

菜品名称

请输入菜品名称

菜品分类

请输入菜品分类

售卖状态

请选择

搜索

2 新建菜品

	菜品名称	图片	菜品分类	售价	售卖状态	最后操作时间	3 操作
<input type="checkbox"/>	鱼香肉丝		荤菜	¥20.00	启售	2019-02-02 11:11	修改 删除 停售
<input type="checkbox"/>	鱼香肉丝		荤菜	¥20.00	停售	2019-02-02 11:11	修改 删除 启售 4
<input type="checkbox"/>	鱼香肉丝		荤菜	¥20.00	启售	2019-02-02 11:11	修改 删除 停售
<input type="checkbox"/>	鱼香肉丝		荤菜	¥20.00	启售	2019-02-02 11:11	修改 删除 停售
<input type="checkbox"/>	鱼香肉丝		荤菜	¥20.00	启售	2019-02-02 11:11	修改 删除 停售

总共 85 个项目 < 1 2 3 4 5 >

菜品的信息：菜品名称、菜品图片、菜品分类、菜品售价、菜品售卖状态、菜品的操作时间(修改时间)。

思考：分类与菜品之间是什么关系？

- 思考逻辑：一个分类下可以有多个菜品吗？反过来再想一想，一个菜品会对应多个分类吗？

答案：一对多关系。一个分类下会有多个菜品，而一个菜品只能归属一个分类。

设计表原则：在多的一方，添加字段，关联属于一这方的主键。

• 页面原型-套餐管理

套餐管理

套餐名称

请输入套餐名称

套餐分类

请输入套餐分类

售卖状态

请选择

搜索

2 新建套餐

<input type="checkbox"/>	套餐名称	套餐图片	套餐分类	套餐价	售卖状态	最后操作时间	3 操作
<input type="checkbox"/>	午餐优惠套餐		午餐套	¥222.00	停售	2019-02-02 11:11	修改 删除 启售 4
<input type="checkbox"/>	午餐优惠套餐		午餐套	¥222.00	启售	2019-02-02 11:11	修改 删除 停售
<input type="checkbox"/>	午餐优惠套餐		午餐套	¥222.00	启售	2019-02-02 11:11	修改 删除 停售
<input type="checkbox"/>	午餐优惠套餐		午餐套	¥222.00	启售	2019-02-02 11:11	修改 删除 停售
<input type="checkbox"/>	午餐优惠套餐		午餐套	¥222.00	启售	2019-02-02 11:11	修改 删除 停售
<input type="checkbox"/>	午餐优惠套餐		午餐套	¥222.00	启售	2019-02-02 11:11	修改 删除 停售

总共 85 个项目

1

2

3

4

5

套餐的信息：套餐名称、套餐图片、套餐分类、套餐价格、套餐售卖状态、套餐的操作时间。

思考：套餐与菜品之间是什么关系？

- 思考逻辑：一个套餐下可以有多个菜品吗？反过来再想一想，一个菜品可以出现在多个套餐中吗？

答案：多对多关系。一个套餐下会有多个菜品，而一个菜品也可以出现在多个套餐中。

设计表原则：创建第三张中间表，建立两个字段分别关联菜品表的主键和套餐表的主键。

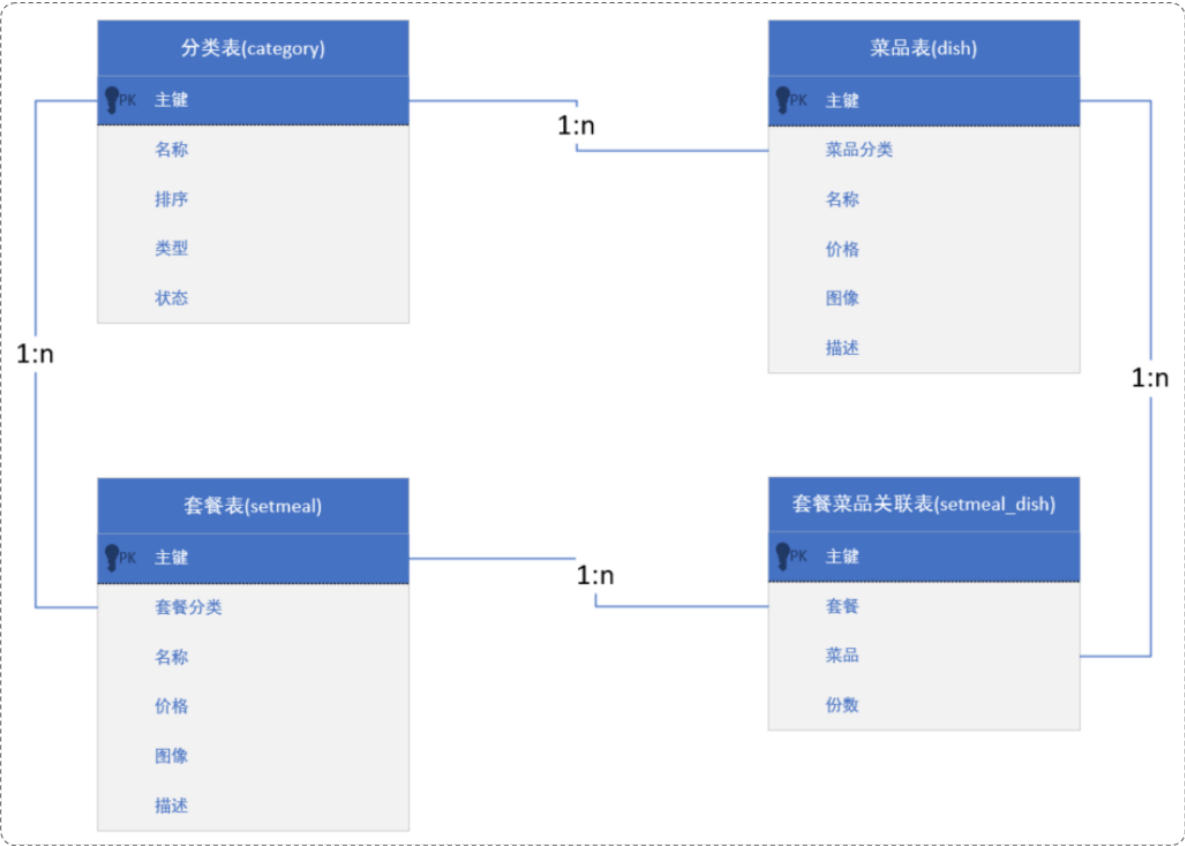
分析页面原型及需求文档后，我们获得：

- 分类表
 - 业务字段：分类名称、分类类型、分类排序、分类状态
 - 基础字段：id(主键)、分类的创建时间、分类的修改时间
- 菜品表
 - 业务字段：菜品名称、菜品图片、菜品分类、菜品售价、菜品售卖状态
 - 基础字段：id(主键)、分类的创建时间、分类的修改时间
- 套餐表
 - 业务字段：套餐名称、套餐图片、套餐分类、套餐价格、套餐售卖状态
 - 基础字段：id(主键)、分类的创建时间、分类的修改时间

表结构之间的关系：

- 分类表 - 菜品表：一对多
 - 在菜品表中添加字段(菜品分类)，关联分类表
- 菜品表 - 套餐表：多对多

- 创建第三张中间表 (套餐菜品关联表)，在中间表上添加两个字段 (菜品id、套餐id)，分别关联菜品表和分类表



表结构

分类表：category

- 业务字段：分类名称、分类类型、分类排序、分类状态
- 基础字段：id (主键)、创建时间、修改时间

1.2.3 新增菜品/套餐名称：
限制字符范围：2-20字符
不符合限制提示：“分类名称输入不符，请输入2-20个字符”

1.2.4 排序不能为空，
内容限制：0-99整数数字
不符合限制提示：“排序只能输入整数类型”
排序为空提示：“排序不能为空”

1.2.5 新增菜品/套餐分类，状态默认为 停用。

1 2. 搜索
2.1 根据分类名称进行搜索，分类名称做模糊搜索
2.2 根据分类类型进行搜索，分类类型做下拉选择 包括菜品分类和套餐分类
2.3 搜索内容为空展示 提示未找到相关分类
2.4 搜索结果页 展示包含搜索关键词的内容。

2 3.分类列表
3.1 排序：根据排序字段顺序做正序排列，如排序字段相同，根据创建时间的倒序排列，单页最多显示10条

3 4.操作
4.1 修改 可以修改分类信息，修改页与新增页布局一致，增加返显数据。
4.2 删除 判断分类下是否有产品
如果有产品提示“分类下有产品不可删除”；
如果没有产品 弹出删除确认窗，确认后删除该分类。

```
1  -- 分类表
2  create table category
3  (
4      id          int unsigned primary key auto_increment comment '主键
                    ID',
5      name        varchar(20)          not null unique comment '分类名称',
6      type        tinyint unsigned not null comment '类型 1 菜品分类 2
                    套餐分类',
7      sort        tinyint unsigned not null comment '顺序',
8      status      tinyint unsigned not null default 0 comment '状态 0
                    禁用, 1 启用',
9      create_time datetime            not null comment '创建时间',
10     update_time datetime            not null comment '更新时间'
11 ) comment '菜品及套餐分类';
```

菜品表：dish

- 业务字段：菜品名称、菜品图片、菜品分类、菜品售价、菜品售卖状态
- 基础字段：id(主键)、分类的创建时间、分类的修改时间

3. 字段限制

3.1 字段限制

字段名称	必填/选填	类型	长度限制	输入限制	是否可重复	提示话术
菜品名称	<u>必填</u>	输入框	<u>2-20</u>	汉字、字母大小写、阿拉伯数字	<u>不可重复</u>	不符合限制提示,“菜品名称输入不符” 菜品重复提示“菜品名称重复, 请调整”
菜品分类	<u>必填</u>	下拉框	-	<u>数据来源分类管理的菜品分类, 根据分类顺序正序排列。</u>	-	-
口味选择	<u>选填</u>	多选框	-	最多不超过4个口味	-	-
价格	<u>必填</u>	输入框	<u>1-8</u>	数字, 可以有 <u>小数点后2位小数</u>	-	不符合限制提示“菜品价格格式有误, 请输入大于零且最多保留两位小数的金额”
图片	<u>必填</u>	图片上传	<u>2M</u>	图片大小不超过2M 仅能上传PNG JPG JPEG类型图片 建议上传200*200或300*300尺寸的图片	-	图片过大, 上传失败 格式错误, 上传失败
菜品描述	<u>选填</u>	输入框	<u>0-200</u>	汉字、字母大小写、阿拉伯数字	-	不符合限制提示,“菜品描述输入不符, 请输入少于200个字”;

```

1  -- 菜品表
2  create table dish
3  (
4      id          int unsigned primary key auto_increment comment '主键
        ID',
5      name        varchar(20)          not null unique comment '菜品名称',
6      category_id int unsigned          not null comment '菜品分类ID',  --
        逻辑外键
7      price        decimal(8, 2)       not null comment '菜品价格',
8      image        varchar(300)         not null comment '菜品图片',
9      description  varchar(200) comment '描述信息',
10     status        tinyint unsigned not null default 0 comment '状态, 0
        停售 1 起售',
11     create_time  datetime              not null comment '创建时间',
12     update_time  datetime              not null comment '更新时间'
13 ) comment '菜品';

```

套餐表: setmeal

- 业务字段: 套餐名称、套餐图片、套餐分类、套餐价格、套餐售卖状态
- 基础字段: id (主键)、分类的创建时间、分类的修改时间

字段名称	必填/选填	类型	长度限制	输入限制	是否可重复	提示话术
套餐名称	<u>必填</u>	输入框	<u>2-20</u>	汉字、字母大小写、阿拉伯数字	<u>不可重复</u>	不符合限制提示, “套餐名称输入不符” 套餐重复提示“套餐名称重复, 请调整”
套餐分类	<u>必填</u>	下拉框	-	数据来源分类管理的套餐分类, 根据分类顺序正序排列。	-	-
价格	<u>必填</u>	输入框	<u>1-8</u>	数字, 可以有小数点后2位小数	-	不符合限制提示“套餐价格格式有误, 请输入大于零且最多保留两位小数的金额”
套餐菜品	<u>必填</u>	多选框	-	-	-	-
图片	<u>必填</u>	图片上传	2M	图片大小不超过2M 仅能上传PNG JPG JPEG类型图片 建议上传200*200或300*300尺寸的图片	-	图片过大, 上传失败 格式错误, 上传失败
套餐描述	<u>选填</u>	输入框	<u>0-200</u>	汉字、字母大小写、阿拉伯数字	-	不符合限制提示, “套餐描述输入不符, 请输入少于200个字”;


```

1  -- 套餐表
2  create table setmeal
3  (
4      id          int unsigned primary key auto_increment comment '主键
        ID',
5      name        varchar(20)          not null unique comment '套餐名称',
6      category_id int unsigned          not null comment '分类id',      --
        逻辑外键
7      price       decimal(8, 2)        not null comment '套餐价格',
8      image       varchar(300)          not null comment '图片',
9      description varchar(200) comment '描述信息',
10     status       tinyint unsigned not null default 0 comment '状态 0:
        停用 1:启用',
11     create_time  datetime              not null comment '创建时间',
12     update_time  datetime              not null comment '更新时间'
13 ) comment '套餐';

```

套餐菜品关联表: setmeal_dish

< 新建套餐

*套餐名称:

*套餐分类:

*套餐价格:

元

*套餐菜品:

1 添加菜品

菜品名称	原价	份数	操作
鱼香肉丝	¥44.00	<div>-</div> <div>1</div> <div>+</div>	删除

2 *套餐图片:

图片大小不超过2M
 仅能上传 PNG JPEG JPG类型图片

套餐描述:

菜品描述, 最长200字

```
1  -- 套餐菜品关联表
2  create table setmeal_dish
3  (
4      id          int unsigned primary key auto_increment comment '主键
ID',
5      setmeal_id int unsigned      not null comment '套餐id ',    -- 逻辑
外键
6      dish_id     int unsigned      not null comment '菜品id',    -- 逻辑
外键
7      copies      tinyint unsigned not null comment '份数'
8  ) comment '套餐菜品关联表';
```