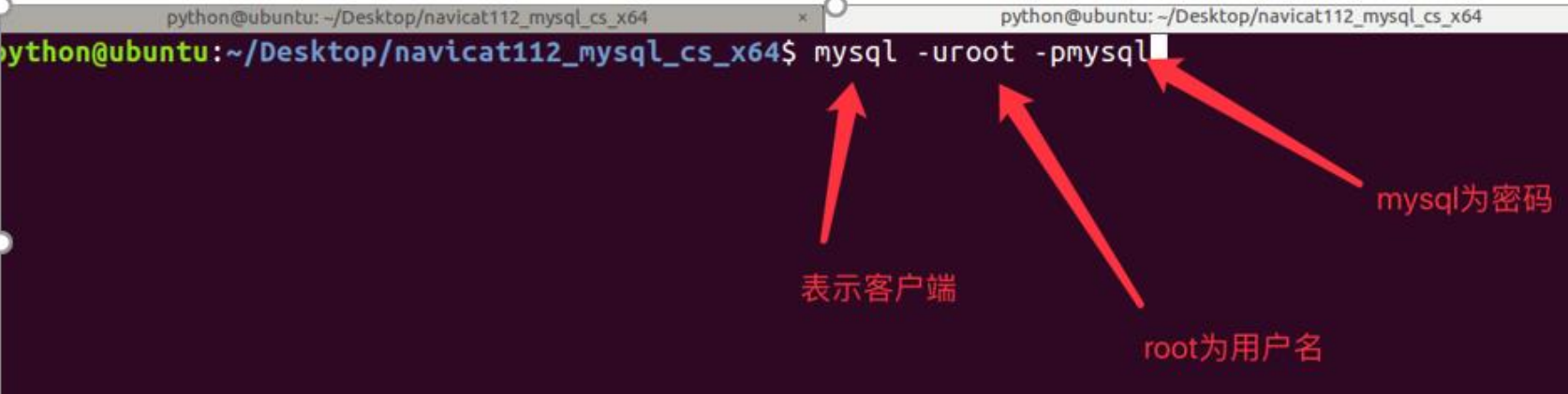


## MySQL 登录



A terminal window showing the command `mysql -uroot -pmysql`. Three red arrows point to parts of the command with Chinese labels: one points to `mysql` with the label "表示客户端" (represents client), another points to `-uroot` with the label "root为用户名" (root is username), and a third points to `-pmysql` with the label "mysql为密码" (mysql is password).

```
python@ubuntu: ~/Desktop/navicat112_mysql_cs_x64
python@ubuntu:~/Desktop/navicat112_mysql_cs_x64$ mysql -uroot -pmysql
```

## MySQL 行和列



A screenshot of a table in a database viewer. The table has 8 columns labeled A through H and 7 rows. The first row contains headers: 'id', 'name', and 'kongfu'. The subsequent rows contain data for five characters: 黄忠 (Huang Zhong), 大乔 (Da Qiao), 小乔 (Xiao Qiao), 貂蝉 (Diao Chan), and 吕布 (Lu Bu). Red text annotations are present: "把列 称之为 字段" (Call columns fields) and "把行 称之为 记录" (Call rows records).

	A	B	C	D	E	F	G	H
1	id	name	kongfu					
2		1 黄忠	炮台					
3		2 大乔	回家					
4		3 小乔	扇子					
5		4 貂蝉	花					
6		5 吕布	方天画戟					
7								

## 课堂笔记-- 数据库的操作

- 链接数据库
- mysql -u 用户名 -p 密码
- mysql -uroot -pmysql
- mysql -uroot -p -- 回车以后 输入密码
  
- 退出数据库
- exit,quit,ctrl-d
  
- sql 语句最后需要有分号;结尾
- 显示数据库版本
- select version();

-- 显示时间

```
select now();
```

-- 查看所有数据库

```
show databases;
```

-- 创建数据库

```
-- create database 数据库名 charset=utf8;
```

```
create database python_24;
```

```
create database python_24_new charset=utf8;
```

-- 查看创建数据库的语句

```
-- show create database ....
```

```
show create database python_24;
```

-- 查看当前使用的数据库

```
select database();
```

-- 使用数据库

```
-- use 数据库的名字
```

```
use python_24_new;
```

-- 删除数据库

```
-- drop database 数据库名;
```

```
drop database python_24;
```

-- 数据表的操作

-- 查看当前数据库中所有表

```
show tables;
```

```
-- 创建表的基本用法
```

```
-- auto_increment 表示自动增长
```

```
-- not null 表示不能为空
```

```
-- primary key 表示主键
```

```
-- default 默认值
```

```
-- create table 数据表名字 (字段 类型 约束[, 字段 类型 约束]);
```

```
-- create table students(字段的名字 类型 约束, 字段 2 的名字 类型 约束);
```

```
-- 创建 students 表(id、name、age、high、gender、cls_id)
```

```
create table students (
```

```
    id int unsigned not null auto_increment primary key,
```

```
    name varchar(30) not null,
```

```
    age tinyint unsigned default 0,
```

```
    high decimal(5,2),
```

```
    gender enum("男", "女", "保密", "第三性别") default "保密",
```

```
    cls_id int unsigned
```

```
);
```

```
-- 作业创建一个 classes 表, 里面有 2 个字段(id name)
```

```
-- 查看表结构
```

```
-- desc 数据表的名字;
```

```
desc students
```

```
-- 查看表的创建语句
```

```
-- show create table 表名字;
```

```
show create table students;
```

```
-- 修改表-添加字段
```

```
-- alter table 表名 add 列名 类型;
```

```
alter table students add birth datetime;
```

```
-- 修改表-修改字段: 不重命名版
```

-- alter table 表名 modify 列名 类型及约束;

alter table students modify birth date;

-- 修改表-修改字段：重命名版

-- alter table 表名 change 原名 新名 类型及约束;

alter table students change birth birthday date default "2000-01-01";

-- 修改表-删除字段

-- alter table 表名 drop 列名;

alter table students drop cls\_id;

-- 删除表

-- drop table 表名;

-- drop database 数据库;

-- drop table 数据表;

-- drop table students;

-- 增删改查(curd)

-- 增加

-- 全列插入

-- insert [into] 表名 values(...)

-- 主键字段 可以用 0 null default 来占位

-- 向 classes 表中插入 一个班级

+-----+-----+-----+-----+-----+-----+		+-----+-----+-----+-----+-----+-----+		+-----+-----+-----+-----+-----+-----+		+-----+-----+-----+-----+-----+-----+		+-----+-----+-----+-----+-----+-----+	
Field	Type	Null	Key	Default		Extra			
+-----+-----+-----+-----+-----+-----+									
id	int(10) unsigned	NO	PRI	NULL		auto_increment			
name	varchar(30)	NO		NULL					
age	tinyint(3) unsigned	YES		0					
high	decimal(5,2)	YES		NULL					
gender	enum('男','女','保密','第三性别')	YES		保密					

```
| birthday | date | YES | | 2000-01-01 | |
+-----+-----+-----+-----+-----+-----+
```

-- 向 students 表插入 一个学生信息

```
insert into students values (0, "曹操", 18, 178.8, "男", "1990-11-11");
insert into students values (0, "曹真", 19, 170.8, "男", "1991-11-11");
insert into students values (null, "曹丕", 19, 170.8, "男", "1991-11-11");
insert into students values (default, "曹植", 19, 170.8, "男", "1991-11-11");
```

-- 失败

```
-- insert into students values(default, "司马懿", 20, 201.1, "第 4 性别", "1990-02-01");
```

-- 枚举中 的 下标从 1 开始 1---"男" 2--->"女"....

```
insert into students values(default, "司马懿", 20, 201.1, 1, "1990-02-01");
insert into students values(default, "小乔", 20, 201.1, 2, "1990-02-01");
```

-- 部分插入

```
-- insert into 表名(列 1,...) values(值 1,...)
insert into students (name, gender) values ("吕布", "男");
```

-- 多行插入

```
insert into students (name, gender) values ("张飞", "男"), ("赵云", "男");
insert into students values (default, "貂蝉", 19, 170.8, "女", "1991-11-11"), (default, "孙尚香", 20, 170.8, "女", "1991-11-11");
```

-- 修改

```
-- update 表名 set 列 1=值 1,列 2=值 2... where 条件;
update students set age=22 where id=7;
```

-- 查询基本使用

-- 查询所有列

```
-- select * from 表名;
select * from students;
```

```
-- 定条件查询
select * from students where gender="女";
select * from students where age<20;

-- 查询指定列
-- select 列 1,列 2,... from 表名;
select name, gender from students;

-- 字段的顺序
select gender, name from students;

-- 可以使用 as 为列或表指定别名
-- select 字段[as 别名], 字段[as 别名] from 数据表 where ....;
select name as 姓名, gender as 性别 from students;

-- 删除
-- 物理删除
-- delete from 表名 where 条件
-- delete from students; -- 这意味着清空数据表的所有数据
delete from students where id=8 or id=9;

-- 逻辑删除
-- 用一个字段来表示 这条信息是否已经不能再使用了
-- 给 students 表添加一个 is_delete 字段 bit 类型
alter table students add is_delete bit(1) default 0;

update students set is_delete=1 where id=6;

select * from students where is_delete=0;
```

## mysql 每日作业

1. 了解关系型数据库的核心元素。

数据行(记录)

数据列(字段)

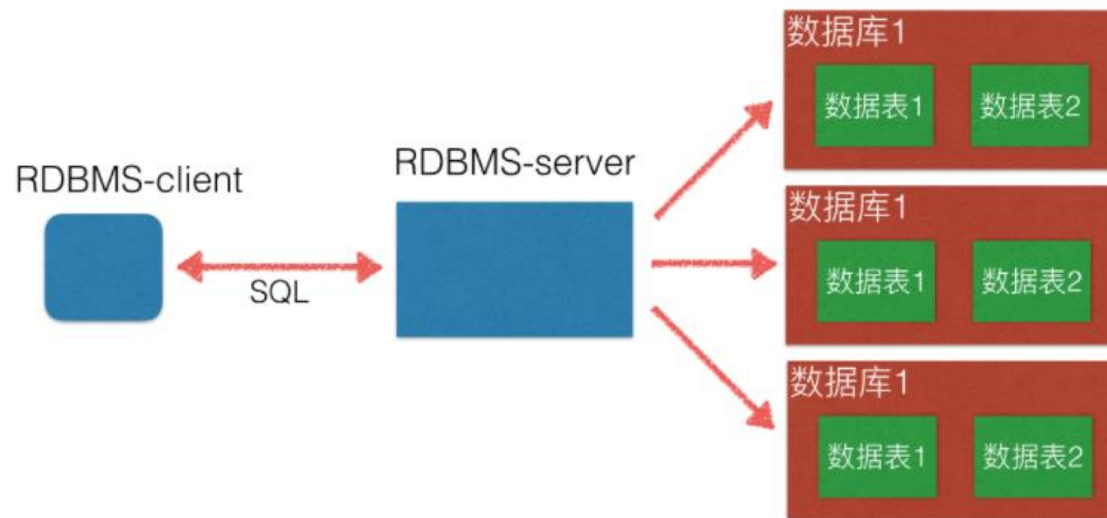
数据表(数据行的集合)

数据库(数据表的集合)

## 2. 什么是 RDBMS?

Relational Database Management System 通过表来表示关系型

## 3. 了解 RMDBS 和数据库的关系。



## 4. 简要说明 mysql 的特点?

### 特点

- 使用 C 和 C++编写，并使用了多种编译器进行测试，保证源代码的可移植性

- 支持多种操作系统，如 Linux、Windows、AIX、FreeBSD、HP-UX、MacOS、NovellNetware、OpenBSD、OS/2 Wrap、Solaris 等
- 为多种编程语言提供了 API，如 C、C++、Python、Java、Perl、PHP、Eiffel、Ruby 等
- 支持多线程，充分利用 CPU 资源
- 优化的 SQL 查询算法，有效地提高查询速度
- 提供多语言支持，常见的编码如 GB2312、BIG5、UTF8
- 提供 TCP/IP、ODBC 和 JDBC 等多种数据库连接途径
- 提供用于管理、检查、优化数据库操作的管理工具
- 大型的数据库。可以处理拥有上千万条记录的大型数据库
- 支持多种存储引擎
- MySQL 软件采用了双授权政策，它分为社区版和商业版，由于其体积小、速度快、总体拥有成本低，尤其是开放源码这一特点，一般中小型网站的开发都选择 MySQL 作为网站数据库
- MySQL 使用标准的 SQL 数据语言形式
- Mysql 是可以定制的，采用了 GPL 协议，你可以修改源码来开发自己的 Mysql 系统
- 在线 DDL 更改功能
- 复制全局事务标识
- 复制无崩溃从机
- 复制多线程从机

5. 完成老师课堂上的数据库操作代码（重点）

- a) 在自己的主机上创建也给数据库，名字叫 qinghua\_db
- b) 在 qinghua\_db 中创建班级表 class,字段包括

id,主键 自增，非空，整型

name 字符串类型（长度 20），非空，

age 整型，默认为 0，非空，

address 字符串类型（长度 200），默认为 NULL 可以为空

- c) 在 class 表中添加 4 条数据，



黄药师，欧阳锋，段智兴，洪七公

+---+-----+---+-----+			
id	name	age	address
+---+-----+---+-----+			
1	黄药师	20	桃花岛
2	欧阳锋	21	白驼山
3	段智星	22	金龙寺
4	洪七公	24	四海为家
+---+-----+---+-----+			

分组

python@ubuntu: ~

女 男 中性 保密

14 rows in set (0.00 sec)

```
mysql> select distinct gender from students;
```

gender
女
男
保密
中性

4 rows in set (0.00 sec)

```
mysql> select gender from student group by gender;
```

ERROR 1146 (42S02): Table 'python\_24\_1.student' doesn't exist

```
mysql> select gender from students group by gender;
```

gender
男
女
中性
保密

4 rows in set (0.01 sec)

mysql>

数据表

分组-2

```
204
265 -- 分组
266 -- group by
267 -- 在students表中, 按照性别分组, 查询所有的性别
268 -- 失败select * from students group by gender;
269 select gender from students group by gender;
270
271 -- 在students表中, 计算每种性别中的人数
272 select gender, count(*) from students group by gender;
273
274 -- 在students表中, 计算男性的人数
275 select count(*) from students where gender=1;
276 select gender, count(*) from students where gender=1 group by gender;
277
278 -- group_concat(...) 1. 先按照gender=1从students表中查询所有复合条件的数据
279 -- 在students表中, 查询男性中的姓名 2. 然后按照gender进行分类
280 3. 再然后对分组进程数据统计
281
282 -- 在students表中, 按照性别分组, 查询平均年龄超过30岁的性别, 以及姓名 having avg(age) > 30
283 -- select gender, group_concat(name), avg(age) from students group by gender having avg(age)>30;
284
285 -- 在students表中, 查询每种性别中的人数多于2个的信息
286
287
288
```

### 分组-3

```
mysql> select gender, avg(age), group_concat(name) from students group by gender having avg(age)>30;
+-----+-----+-----+
| gender | avg(age) | group_concat(name) |
+-----+-----+-----+
| 男     | 32.6000  | 彭于晏,刘德华,周杰伦,程坤,郭靖 |
| 中性   | 33.0000  | 金星                 |
+-----+-----+-----+
2 rows in set (0.00 sec)

mysql>
```

如果判断的条件放在了group by的后面, 那么意味着  
1. 先按照之前的所有的分组方式进行分组查询  
2. 对分组查询出来的结果按照 此时的条件进行过滤

### 自关联

## js实现全国三级城市联动select选择

省份

id	name
1	北京市
2	山东省
3	河北省
4	山西省
5	安徽省

市

id	name	p_id
1	北京市	1
2	济南市	2
3	青岛市	2
4	烟台市	2

区县

id	name	xxx_id
1	朝阳区	1
2	崂山区	3
3	市南区	3
3	市北区	3
4	黄岛区	3

大队

村

乡镇

## js实现全国三级城市联动select选择

山东省

青岛市

黄岛区

id	name	p_id
1	北京市	null
2	山东省	null
3	河北省	null
4	青岛市	2
5	烟台市	2
	崂山区	4

### 每日作业

- 学生表结构设计为：姓名、生日、性别、家乡，并且学生表与班级表为多对一的关系，写出创建学生表的语句

```
create table students(  
id int unsigned auto_increment primary key not null,  
name varchar(10) not null,  
birthday date,  
gender bit default 1,  
hometown varchar(20),  
clsid int unsigned,  
isdelete bit default 0,
```

```
foreign key(clsid) references classes(id)
);
```

- 向学生表中插入数据：
  - **python1** 班有郭靖、黄蓉，要求：使用全列插入，一次一值
  - **python2** 班有杨过、小龙女，要求：使用指定列插入，一次一值
  - 未分班的学生有黄药师、洪七公、洪七婆，要求：使用指定列插入，一次多值

```
insert into students values(0,'郭靖','2016-1-1',1,'蒙古',1,0);
insert into students values(0,'黄蓉','2016-5-8',0,'桃花岛',1,0);
insert into students(name,gender,clsid) values('杨过',1,2);
insert into students(name,clsid) values('小龙女',2);
insert into students(name) values('黄药师'),('洪七公'),('洪七婆');
```

- 设计科目表 **subjects**，包括科目名称

```
create table subjects(
id int unsigned auto_increment primary key not null,
name varchar(20),
isdelete bit default 0
);
```

- 向表中插入数据 **python**、数据库、前端

```
insert into subjects(name) values('python'),('数据库'),('前端');
```

- 设计成绩表，字段包括：学生、科目、成绩

```
create table scores(
id int unsigned auto_increment primary key not null,
score int,
stuid int unsigned,
subid int unsigned,
foreign key(stuid) references students(id),
foreign key(subid) references subjects(id)
```

```
);
```

- 向学生表中添加一些示例数据

```
insert into scores(score,stuid,subid) values
(100,1,1),(98,1,2),(90,1,3),
(95,2,1),(100,2,2),(98,2,3),
(90,3,1),(80,3,2),(85,3,3),
(70,4,1),(60,4,2),(87,4,3);
```

- 查询学生表的所有行、所有列数据
- 查询学生表的所有行数据，只显示编号、姓名列
- 查询学生表的所有家乡数据，消除重复行
- 查询编号不大于 4 的科目
- 查询没被删除的学生
- 查询编号大于 3 的女同学
- 查询姓黄的女学生
- 查询学号是 3 至 8 的学生
- 查询没有填写地址的学生
- 查询未删除的男生总人数
- 查询各城市人数
- 查询未删除男生信息，按学号降序
- 查询前 3 行男生信息

- 查询学生姓名及班级名称

## 每日测试

```
-- 创建 "京东" 数据库
create database jing_dong charset=utf8;

-- 使用 "京东" 数据库
use jing_dong;

-- 创建一个商品 goods 数据表
create table goods(
    id int unsigned primary key auto_increment not null,
    name varchar(150) not null,
    cate_name varchar(40) not null,
    brand_name varchar(40) not null,
    price decimal(10,3) not null default 0,
    is_show bit not null default 1,
    is_saleoff bit not null default 0
);
```

## 插入数据

```
-- 向 goods 表中插入数据

insert into goods values(0,'r510vc 15.6 英寸笔记本','笔记本','华硕','3399',default,default);
insert into goods values(0,'y400n 14.0 英寸笔记本电脑','笔记本','联想','4999',default,default);
insert into goods values(0,'g150th 15.6 英寸游戏本','游戏本','雷神','8499',default,default);
insert into goods values(0,'x550cc 15.6 英寸笔记本','笔记本','华硕','2799',default,default);
insert into goods values(0,'x240 超极本','超极本','联想','4880',default,default);
```



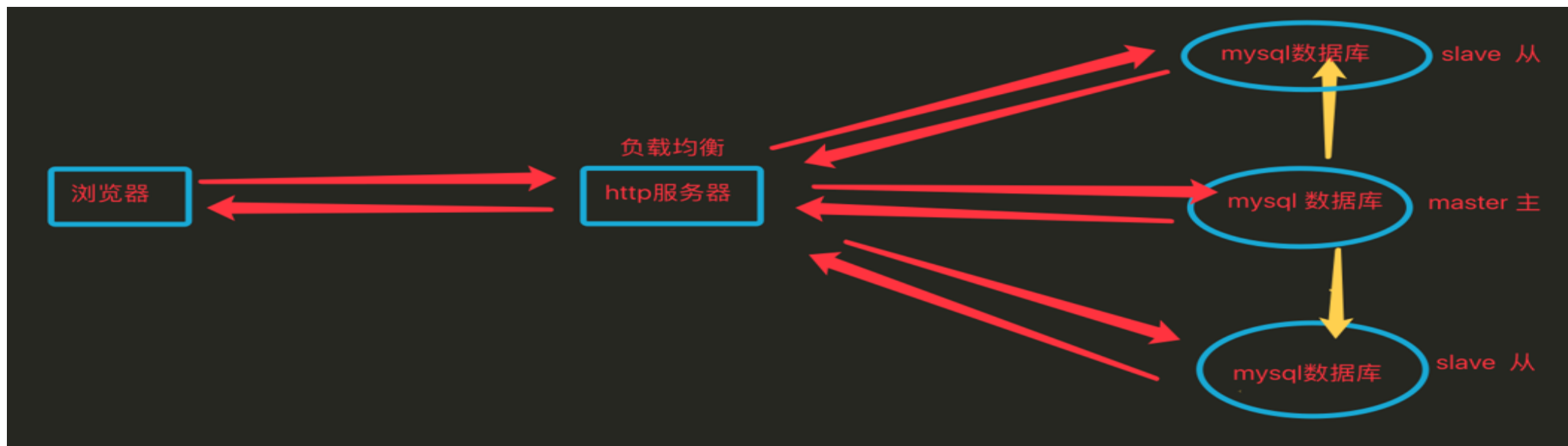
```

insert into goods values(0,'u330p 13.3 英寸超极本','超级本','联想','4299',default,default);
insert into goods values(0,'svp13226scb 触控超极本','超级本','索尼','7999',default,default);
insert into goods values(0,'ipad mini 7.9 英寸平板电脑','平板电脑','苹果','1998',default,default);
insert into goods values(0,'ipad air 9.7 英寸平板电脑','平板电脑','苹果','3388',default,default);
insert into goods values(0,'ipad mini 配备 retina 显示屏','平板电脑','苹果','2788',default,default);
insert into goods values(0,'ideacentre c340 20 英寸一体电脑 ','台式机','联想','3499',default,default);
insert into goods values(0,'vostro 3800-r1206 台式电脑','台式机','戴尔','2899',default,default);
insert into goods values(0,'imac me086ch/a 21.5 英寸一体电脑','台式机','苹果','9188',default,default);
insert into goods values(0,'at7-7414lp 台式电脑 linux ）、','台式机','宏碁','3699',default,default);
insert into goods values(0,'z220sff f4f06pa 工作站','服务器/工作站','惠普','4288',default,default);
insert into goods values(0,'poweredge ii 服务器','服务器/工作站','戴尔','5388',default,default);
insert into goods values(0,'mac pro 专业级台式电脑','服务器/工作站','苹果','28888',default,default);
insert into goods values(0,'hmz-t3w 头戴显示设备','笔记本配件','索尼','6999',default,default);
insert into goods values(0,'商务双肩背包','笔记本配件','索尼','99',default,default);
insert into goods values(0,'x3250 m4 机架式服务器','服务器/工作站','ibm','6888',default,default);
insert into goods values(0,'商务双肩背包','笔记本配件','索尼','99',default,default);

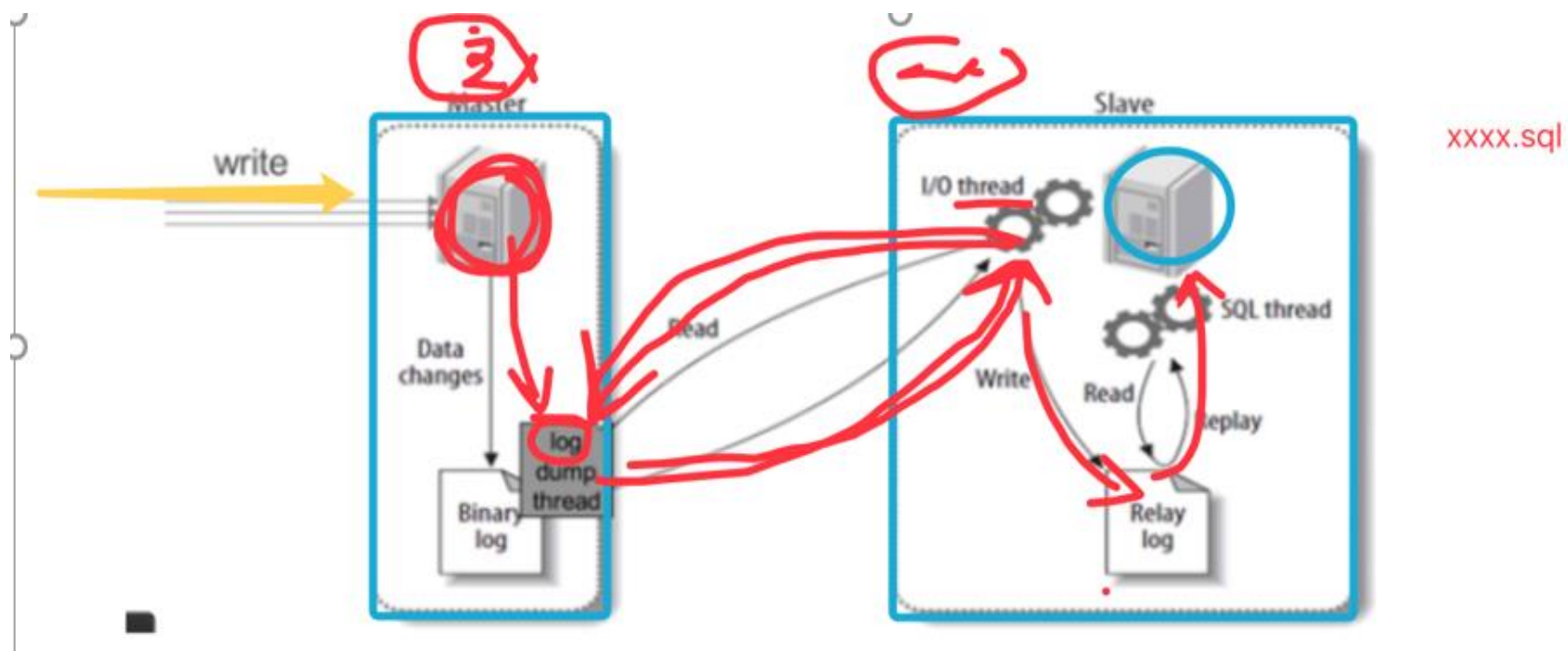
```

- 查询类型 cate\_name 为 '超极本' 的商品名称、价格
- 显示商品的种类
- 求所有电脑产品的平均价格,并且保留两位小数
- 显示每种商品的平均价格
- 查询每种类型的商品中 最贵、最便宜、平均价、数量
- 查询所有价格大于平均价格的商品，并且按价格降序排序
- 查询每种类型中最贵的电脑信息

mysql 主从



mysql 主从-2



```
python@ubuntu: ~/Desktop/python24/16day
python@ubuntu:~/Desktop/python24/16day$ ls
01-查询数据.py
python@ubuntu:~/Desktop/python24/16day$ python3 01-查询数据.py
请输入商品名字 " or 1=1 or "
select * from goods where name=" or 1=1 or ";
((1, 'r510vc 15.6英寸笔记本', 5, 2, Decimal('3399.000'), b'\x01', b'\x00'), (2, 'y400n 14.0英寸笔记本电脑', 5, 7, Decimal('4999.000'), b'\x01', b'\x00'), (3, 'g150th 15.6英寸游戏本', 4, 9, Decimal('8499.000'), b'\x01', b'\x00'), (4, 'x550cc 15.6英寸笔记本', 5, 2, Decimal('2799.000'), b'\x01', b'\x00'), (5, 'x240 超极本', 7, 7, Decimal('4880.000'), b'\x01', b'\x00'), (6, 'u330p 13.3英寸超极本', 7, 7, Decimal('4299.000'), b'\x01', b'\x00'), (7, 'svp13226scb 触控超极本', 7, 6, Decimal('7999.000'), b'\x01', b'\x00'), (8, 'ipad mini 7.9英寸平板电脑', 2, 8, Decimal('1998.000'), b'\x01', b'\x00'), (9, 'ipad air 9.7英寸平板电脑', 2, 8, Decimal('3388.000'), b'\x01', b'\x00'), (10, 'ipad mini 配备 retina 显示屏', 2, 8, Decimal('2788.000'), b'\x01', b'\x00'), (11, 'ideacentre c340 20英寸一体机', 1, 7, Decimal('3499.000'), b'\x01', b'\x00'), (12, 'vostro 3800-r1206 台式电脑', 1, 5, Decimal('2899.000'), b'\x01', b'\x00'), (13, 'imac me086ch/a 21.5英寸一体机', 1, 8, Decimal('9188.000'), b'\x01', b'\x00'), (14, 'at7-7414lp 台式电脑 linux', 1, 3, Decimal('3699.000'), b'\x01', b'\x00'), (15, 'z220s ff f4f06pa工作站', 3, 4, Decimal('4288.000'), b'\x01', b'\x00'), (16, 'poweredge ii服务器', 3, 5, Decimal('5388.000'), b'\x01', b'\x00'), (17, 'mac pro专业级台式电脑', 3, 8, Decimal('28888.000'), b'\x01', b'\x00'), (18, 'hmr-t3w 头戴显示设备', 6, 6, Decimal('6999.000'), b'\x01', b'\x00'), (19, '商务双肩背包', 6, 6, Decimal('99.000'), b'\x01', b'\x00'), (20, 'x3250 m4 机架式服务器', 3, 1, Decimal('6888.000'), b'\x01', b'\x00'), (21, '商务双肩背包', 6, 6, Decimal('99.000'), b'\x01', b'\x00'), (24, '鼠标', 3, 8, Decimal('535.000'), b'\x01', b'\x00'))
python@ubuntu:~/Desktop/python24/16day$
```

## 第二范式

满足第二范式并且每个字段都不间接依赖于主键列。

```
CREATE TABLE province
(
    pr_id UNSIGNED INT NOT NULL AUTO_INCREMENT, -- 主键
    pr_name VARCHAR(20) NOT NULL, -- 省份名, 完全依赖于主键, pr_id 定了, pr_name 就定了
    PRIMARY KEY(pr_id)
);

CREATE TABLE city
(
    ct_id UNSIGNED INT NOT NULL AUTO_INCREMENT, -- 主键
    ct_name VARCHAR(20) NOT NULL, -- 完全依赖于主键, ct_id 定了, ct_name 就定了
    pr_id UNSIGNED INT NOT NULL, -- 完全依赖于主键, ct_id 定了, 就可以确定 pr_id
    pr_name VARCHAR(20) NOT NULL, -- 完全依赖于主键, ct_id 定了, 就可以确定 pr_name
    PRIMARY KEY(ct_id),
    FOREIGN KEY(pr_id) REFERENCES province(pr_id) ON DELETE CASCADE
);
```

3 山东省

1 青岛 3 山东省  
2 菏泽 3 山东省

上述的这两张表都满足第二范式，不过，注意到 city 表中的 pr\_name 字段虽然完全依赖于 ct\_id，但是它



举个例子，美国销售军火的时候，对每一样武器，根据国家或地区的不同而给出不同的价格。建个表看看：

```
CREATE TABLE weapon_price
(
    wp_id UNSIGNED INT NOT NULL AUTO_INCREMENT, -- 武器编号
    cs_id UNSIGNED INT NOT NULL, -- 消费者 id
    wp_price UNSIGNED INT NOT NULL, -- 武器价格，根据武器买主的不同而不同
    cs_name VARCHAR(40) NOT NULL -- 消费者的称呼，例如 菲律宾/韩国
);
```

1	10	10000000	韩国
2	10	98129381	韩国
4	10	10000000	韩国

weapon\_price 用于描述武器的价格，价格根据(武器，消费者)的不同而不同。

对于此表 (wp\_id,cs\_id) 是其主键。其中 wp\_price 是完全依赖于 (wp\_id,cs\_id) 的，而 cs\_name 则只依赖于 cs\_id，即只依赖于主键的一部分。

这种情况导致的问题是什么呢？

	A	B	C	D	E	F	G	H	I	J	K	L
1		商品表-goods							冗余			
2	id	name	cate_id	brand_id	price_id	is_show	is_saleoff					
3	8001	mac book pro 15吋	电脑	Apple	18888	1	0					
4			台式机	联想								
5			电脑									
6			台式机									
7		商品分类表-goods_cates										
8	id	name						商品表-goods				
9	401	电脑						id	name	cate_name	brand_name	price
10	402	台式机						8001	mac book pro 15吋	笔记本	苹果	18888
11								8002	mac book pro 15吋	笔记本	苹果	18888
12								8003	mac book pro 15吋	笔记本	苹果	18888
13		商品品牌表-goods_brands						8004	mac book pro 15吋	笔记本	苹果	18888
14	id	name						8005	mac book pro 15吋	笔记本	苹果	18888
15	701	Apple										
16	701	联想										
17												
18												

update goods as g inner join goods\_cates as c on g.cate\_name=c.name set g.cate\_id=c.id;

insert into 表名 values (对应的字段的数据);  
insert into 表名 (字段名) values (对应的字段的数据);

- 将分组结果写入到goods\_cates数据表

```
insert into goods_cates (name) select cate_name from goods group by cate_name;
```

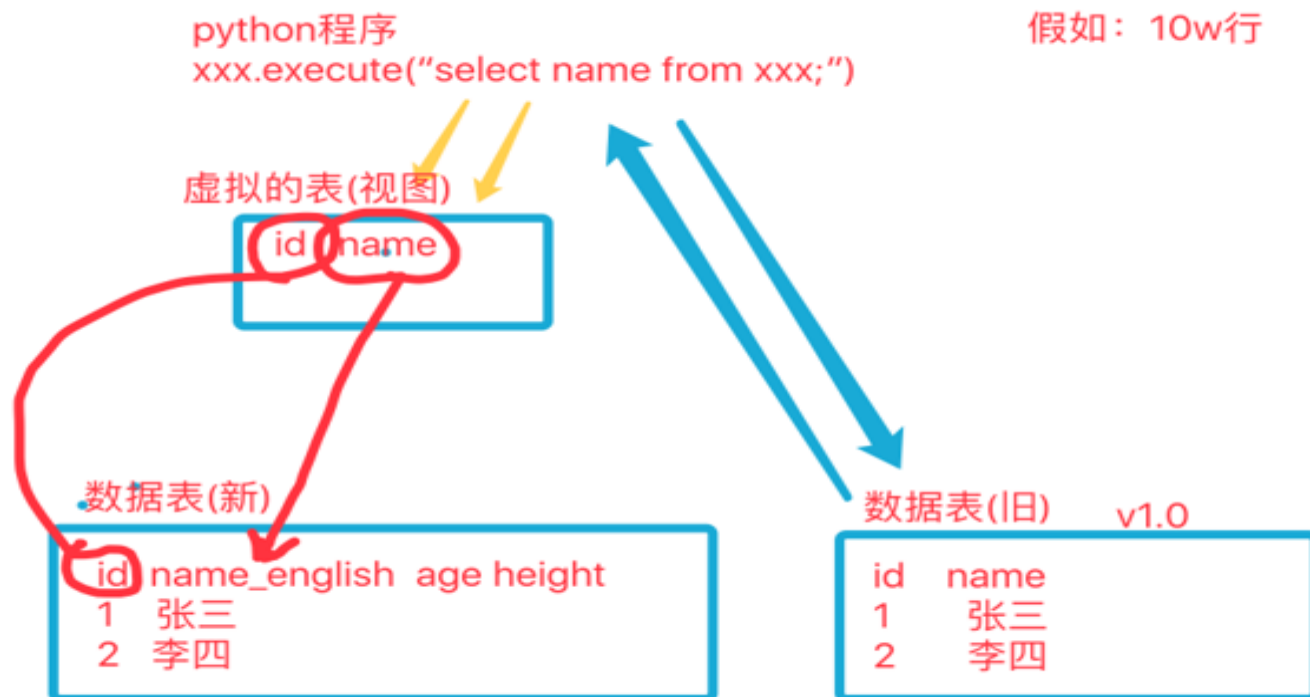
```
python@ubuntu:~$ mysql -u root -p
mysql> select * from goods as g inner join (select cate_name, max(price) as max_price from goods group by cate_name) as a
on a.cate_name=g.cate_name
and a.max_price=g.price;
7 rows in set (0.01 sec)
```

cate_name	max_price
台式机	9188.000
平板电脑	3388.000
服务器/工作站	28888.000
游戏本	8499.000
笔记本	4999.000
笔记本配件	6999.000
超级本	7999.000

```
mysql> select * from goods
where a.cate_name=g.cate_name
and a.max_price=g.price;
21 rows in set (0.00 sec)
```

id	name	cate_name	brand_name	price	is_show	is_saleoff
1	r510vc 15.6英寸笔记本	笔记本	华硕	3399.000	0001	
2	y400n 14.0英寸笔记本电脑	笔记本	联想	4999.000	0001	
3	g150th 15.6英寸游戏本	游戏本	雷神	8499.000	0001	
4	x550cc 15.6英寸笔记本	笔记本	华硕	2799.000	0001	
5	x240 超极本	超级本	联想	4880.000	0001	
6	u330p 13.3英寸超极本	超级本	联想	4299.000	0001	
7	svp13226scb 触控超极本	超级本	索尼	7999.000	0001	
8	ipad mini 7.9英寸平板电脑	平板电脑	苹果	1998.000	0001	
9	ipad air 9.7英寸平板电脑	平板电脑	苹果	2299.000	0001	
10	ipad mini 配备 retina 显示屏	平板电脑	苹果	2788.000	0001	
11	ideacentre c340 20英寸一体电脑	台式机	联想	3499.000	0001	
12	vostro 3800-r1206 台式电脑	台式机	戴尔	2899.000	0001	
13	imac me086ch/a 21.5英寸一体电脑	台式机	苹果	6188.000	0001	
14	at/-/414lp 台式电脑 (linux )	台式机	宏碁	3699.000	0001	
15	z220sff f4f06pa 工作站	服务器/工作站	惠普	4288.000	0001	
16	poweredge i1服务器	服务器/工作站	戴尔	5388.000	0001	
17	mac pro专业级台式电脑	服务器/工作站	苹果	28888.000	0001	
18	hmz-t3w 头戴显示设备	笔记本配件	索尼	6999.000	0001	
19	商务双肩背包	笔记本配件	索尼	99.000	0001	
20	x3250 m4机架式服务器	服务器/工作站	ibm	6888.000	0001	
21	商务双肩背包	笔记本配件	索尼	99.000	0001	

视图



## 索引

```
mysql> show profiles;
```

Query_ID	Duration	Query
1	0.03839250	select * from test_index where title='ha-99999'
2	0.16615675	create index title_index on test_index(title(10))
3	0.00066225	select * from test_index where title='ha-99999'

```
3 rows in set, 1 warning (0.00 sec)
```

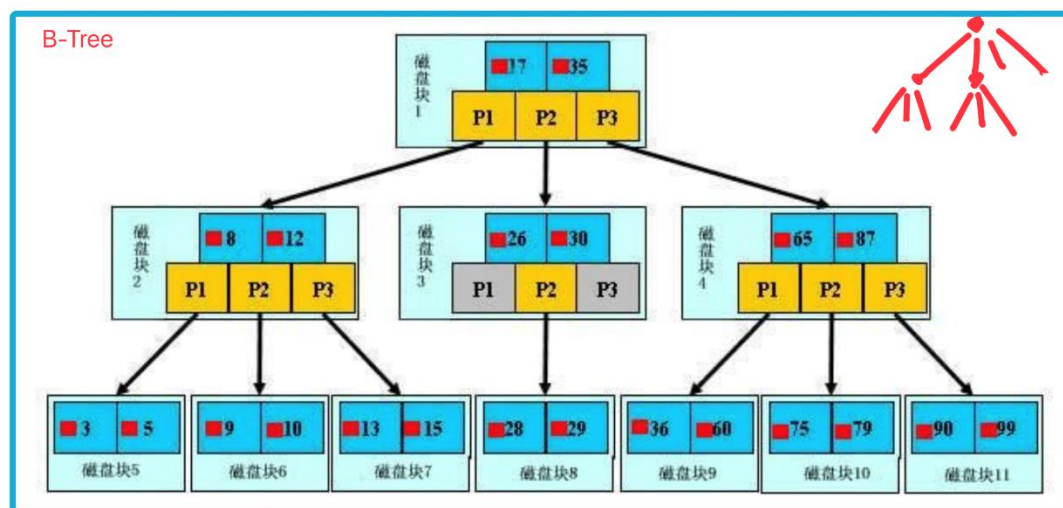
```
mysql>
```

对经常需要查询的字段添加索引之后  
能够大大的提升查询的效率

## 索引的原理

数据库也是一样，但显然要复杂许多，因为不仅面临着等值查询，还有范围查询(>、<、between、in)、模糊查询(like)、并集查询(or)等等。数据库应该选择什么样的方式来应对所有的问题呢？我们回想字典的例子，能不能把数据分成段，然后分段查询呢？最简单的如果1000条数据，1到100分成第一段，101到200分成第二段，201到300分成第三段.....这样查第250条数据，只要找第三段就可以了，一下子去除了90%的无效数据。

`select * from xxx where num=76;`



id	num	.....
1	30	
2	60	
3	20	
4	11	
5	55	
6	33	
7	78	

代码 code

## 01- 查询数据

```
import pymysql
```

```
def main():
```

```
    """获取用户需要查询的信息，并且显示"""
```

```
    # 1. 获取用户需要查询的商品名字
```

```
    find_item_name = input("请输入商品名字:")
```

```
    # 2. 链接数据库
```

```
    conn = pymysql.connect(host='localhost',port=3306,database='jing_dong',user='root',password='mysql',charset='utf8')
```

```
    # 3. 获取游标对象
```

```

cursor = conn.cursor()

# 4. 组织字符串
sql = """select * from goods where name="%s";""" % find_item_name
print(sql)

# 5. 执行查询语句
cursor.execute(sql)

# 6. 显示相应的结果
print(cursor.fetchall())

# 7. 关闭
cursor.close()
conn.close()

if __name__ == "__main__":
    main()

```

## 02- 查询数据-防止 sql 注入

```

import pymysql

def main():
    """获取用户需要查询的信息，并且显示"""
    # 1. 获取用户需要查询的商品名字
    find_item_name = input("请输入商品名字:")

    # 2. 链接数据库
    conn = pymysql.connect(host='localhost',port=3306,database='jing_dong',user='root',password='mysql',charset='utf8')

    # 3. 获取游标对象
    cursor = conn.cursor()

```



# 4. 组织字符串

```
# sql = """select * from goods where name="%s";""" % find_item_name
```

```
# print(sql)
```

```
sql = """select * from goods where name="%s";"""
```

# 5. 执行查询语句

```
cursor.execute(sql, [find_item_name])
```

# 6. 显示相应的结果

```
print(cursor.fetchall())
```

# 7. 关闭

```
cursor.close()
```

```
conn.close()
```

```
if __name__ == "__main__":
```

```
    main()
```