**Mysql5.7.20**

# 1.安装

## 1.1 下载安装文件

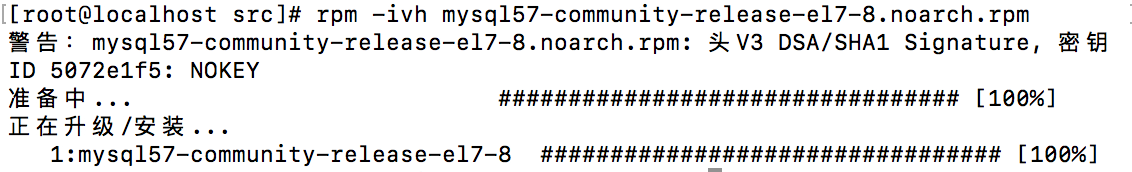
wget http://repo.mysql.com/mysql57-community-release-el7-8.noarch.rpm

cd /usr/local/src 文件下



## 1.2 执行rpm

rpm -ivh mysql57-community-release-el7-8.noarch.rpm



## 1.3 安装

yum -y install mysql-server

默认配置文件路径：

配置文件：/etc/my.cnf

日志文件：/var/log/var/log/mysqld.log

服务启动脚本：/usr/lib/systemd/system/mysqld.service

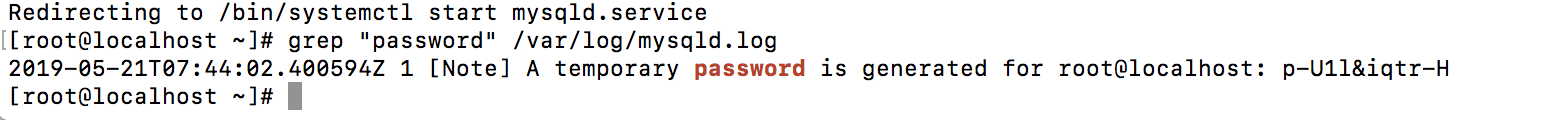
socket文件：/var/run/mysqld/mysqld.pid

## 1.4 启动

### 1.4.1 查看密码

service mysqld restart

查看初始密码：grep “password” /var/log/mysqld.log

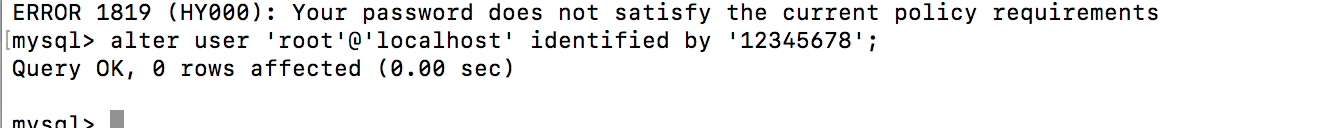


### 1.4.2 登陆并修改密码

mysql -u root -p

输入密码登陆

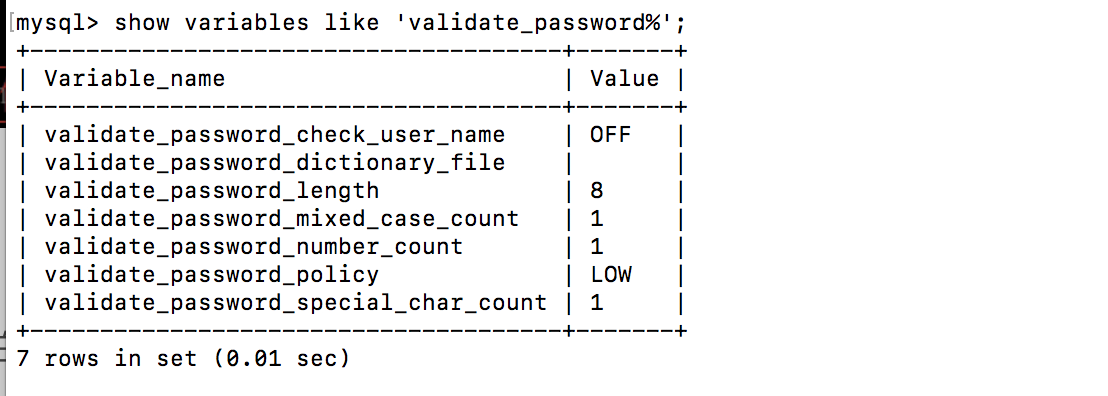
首次登陆需要修改密码，



注意：以上是修改成功的，5.7版本的对密码有要求，要求的登记默认是MEDIUM（中等），需要把这个等级设置为LOW，密码长度也可以设置(默认是8)，做完如下设置后，执行修改方能修改为简单的密码

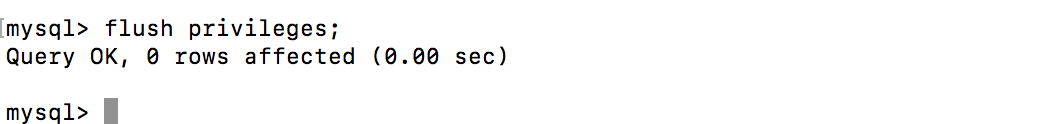
set global validate\_password\_policy=LOW;

set global validate\_password\_length=6;



最后，执行

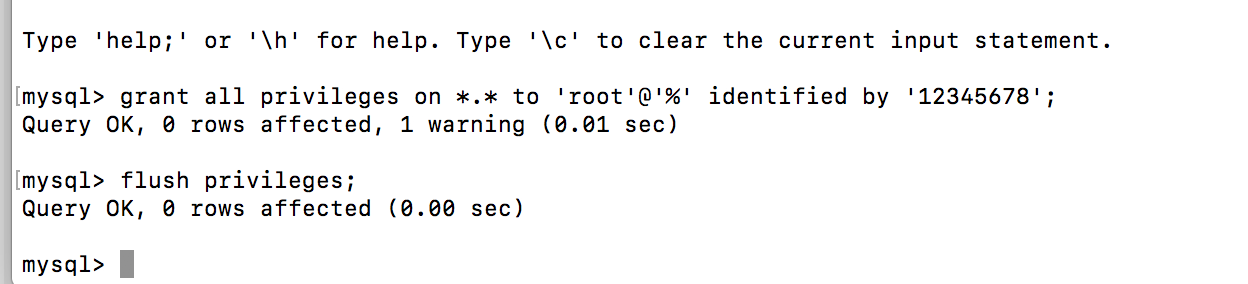
flush privileges;



### 1.4.3 设置远程连接

Grant all privileges on \*.\* to ‘root’@’%’ identified by ‘password’;

Flush privileges;



# 2.性能优化

## 2.1 mysql版本

### 2.1.1 版本介绍

5.4-5.x：mysql整合了三方公司的新存储引擎（推荐5.5）

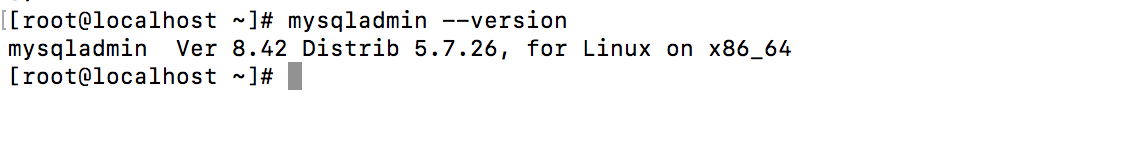
### 2.1.2 安装

略

查看版本

#### 2.1.2.1 查看版本

mysqladmin --version



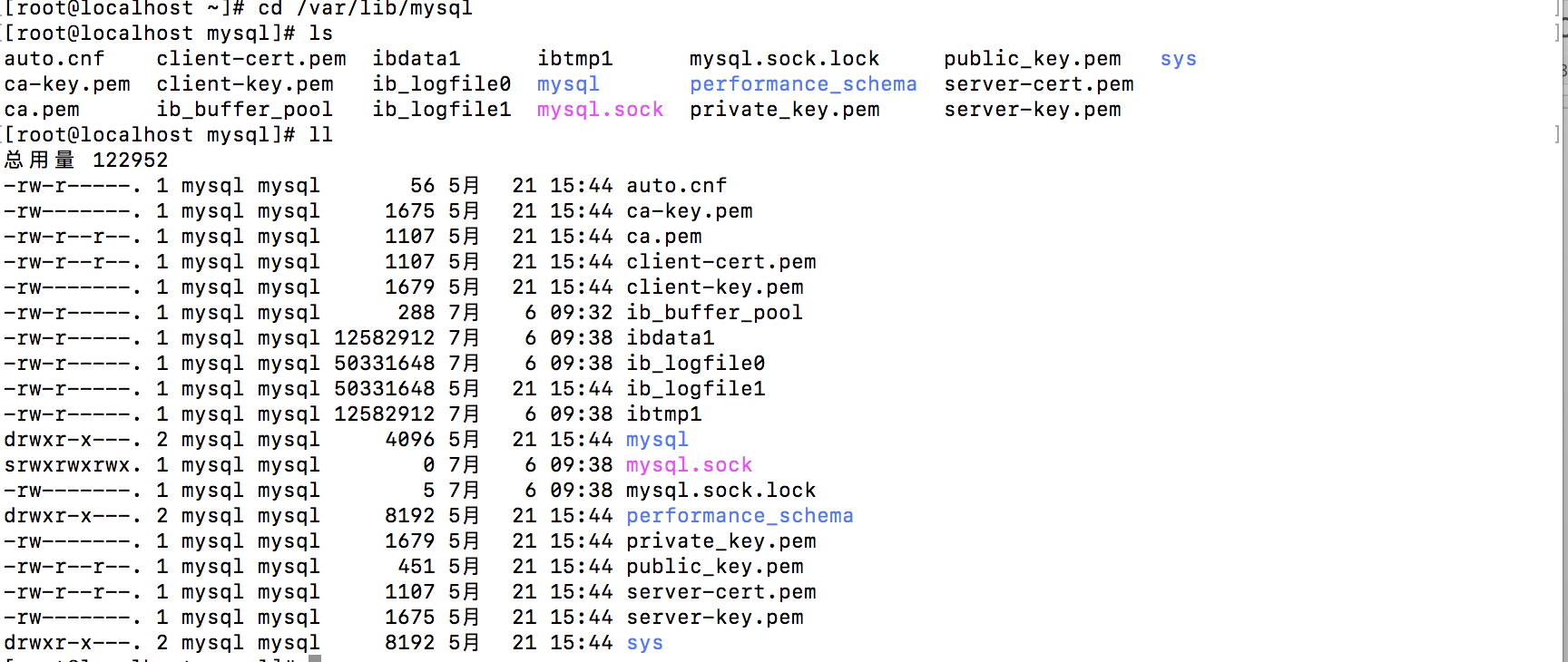
#### 2.1.2.2 启动，关闭

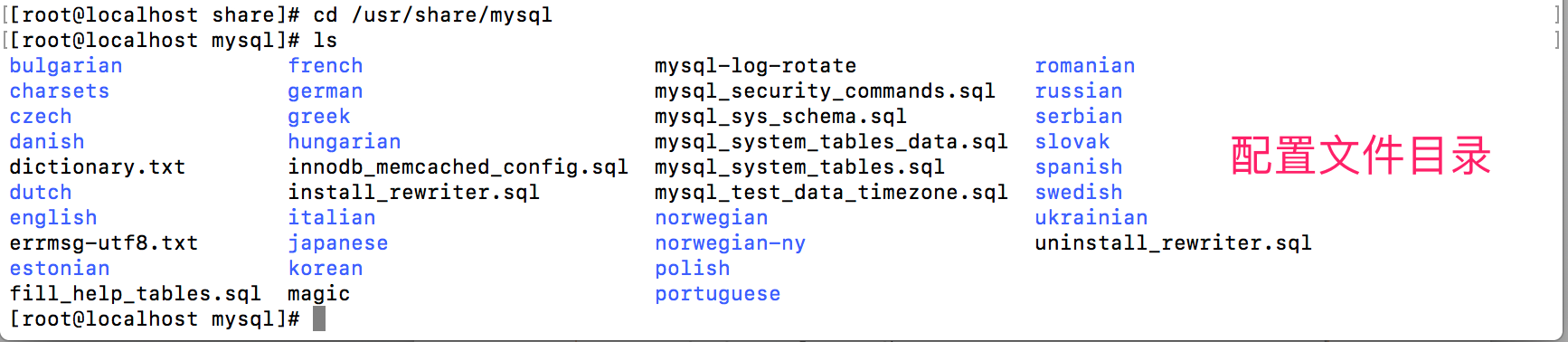


#### 2.1.2.3 查找mysql

ps –ef|grep myql







mysql 命令

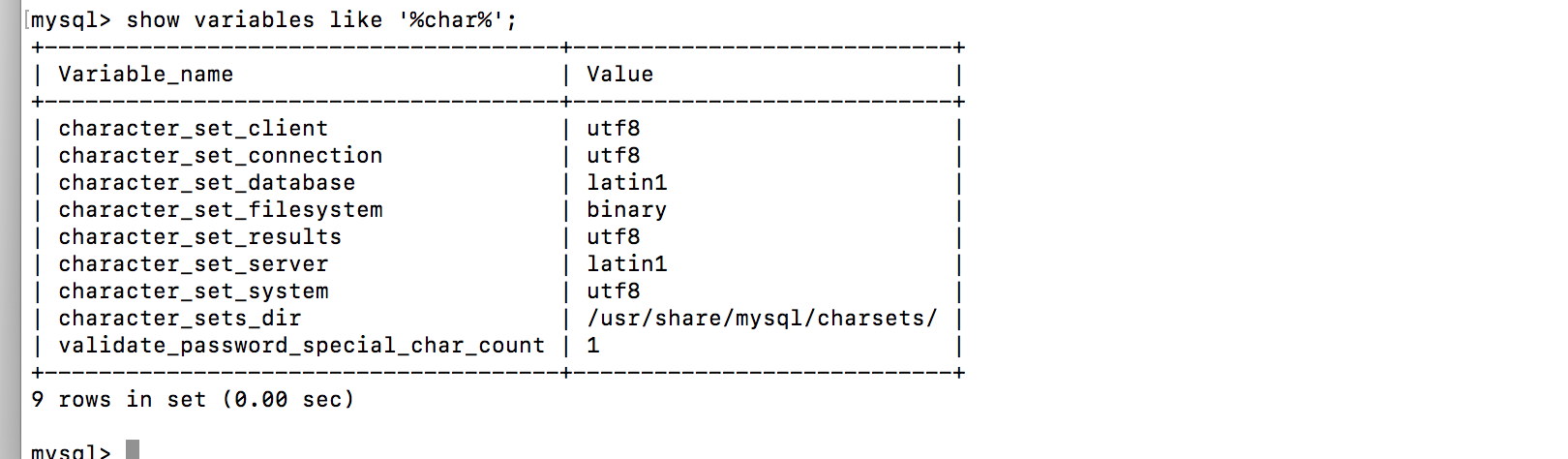


#### 2.1.2.4 mysql脚本及配置文件

/etc/my.cnf

#### 2.1.2.5 mysql字符编码

show variables like ‘%char%’



统一设置编码为utf-8，改配置文件,版本和视频版本不一致，操作不一样

system clear 清屏命令

### 2.1.3 卸载

yum –y remove mysql-libs-5.1.73-7.el6\*

## 2.2 mysql底层原理

### 2.2.1 mysql分层

连接层：提供与客户端连接的服务

服务层：1提供各种用户使用的接口（select）

2提供sql优化器（MYSQL Query Optimizer）

引擎层：各种存储数据的方式（innoDB，MyISAM）

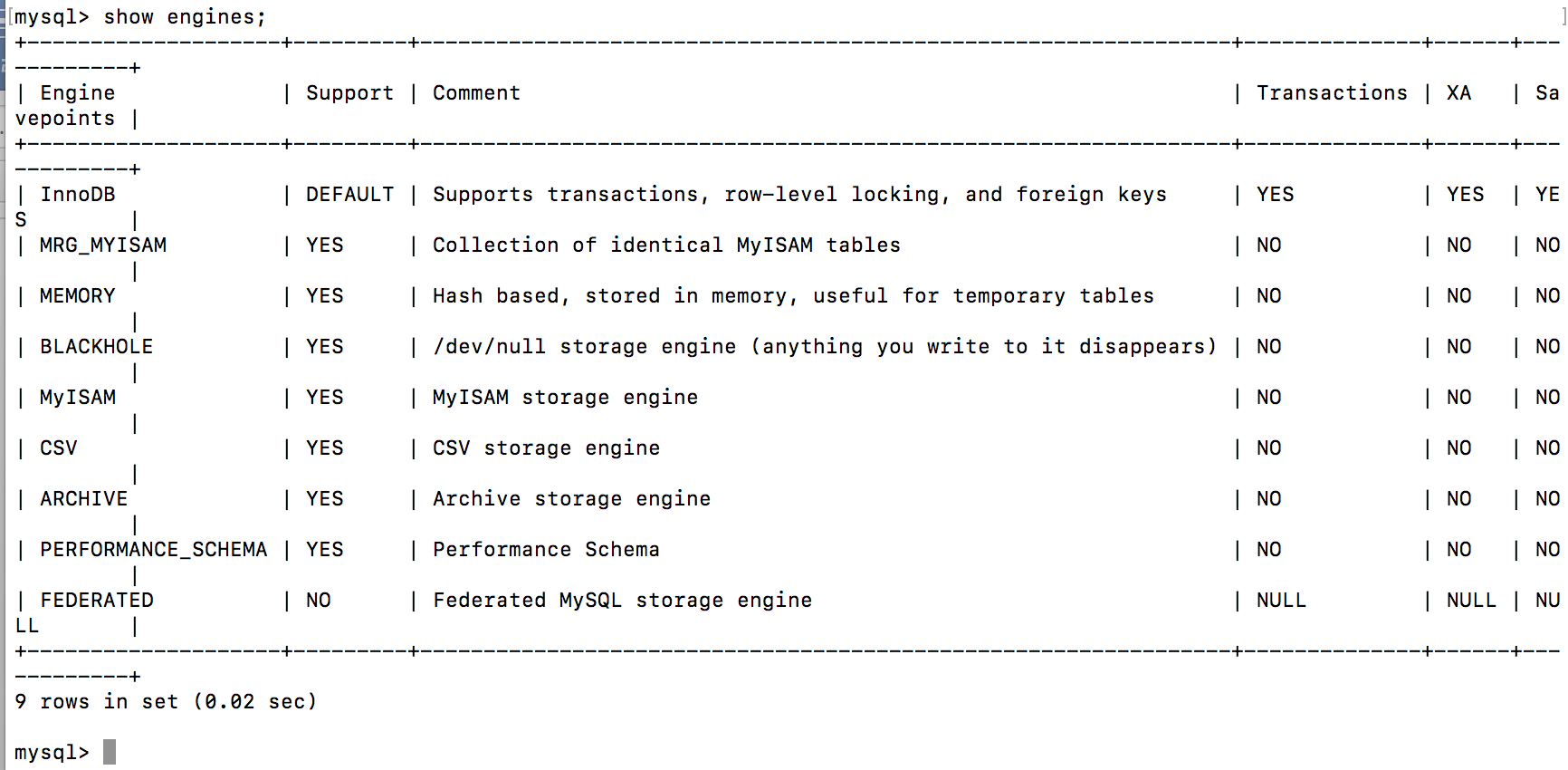
innoDB，MyISAM主要区别：

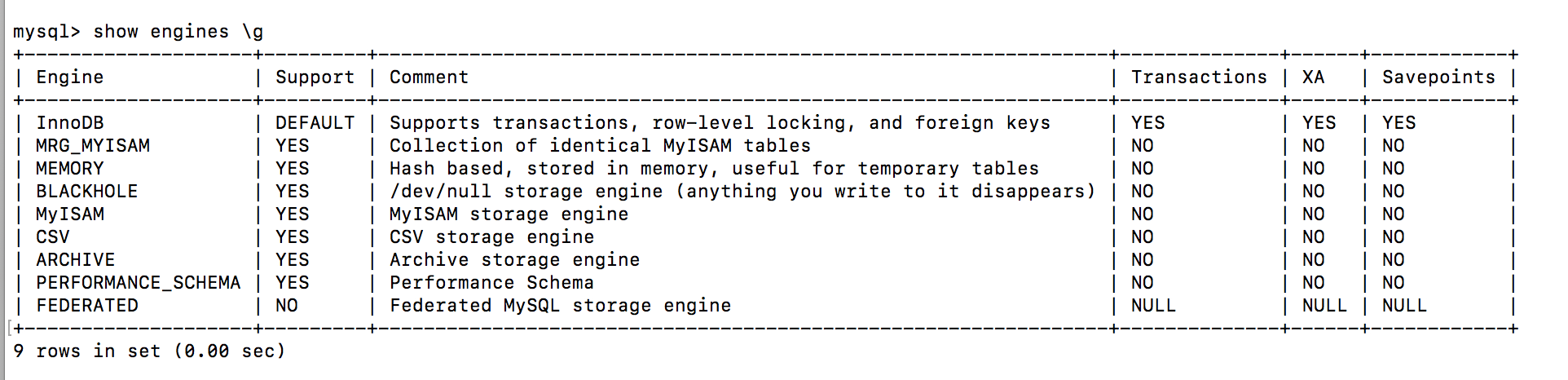
innoDB事务优先，（适合高并发），行锁

MyISAM：性能优先，表锁

存储层：存储数据

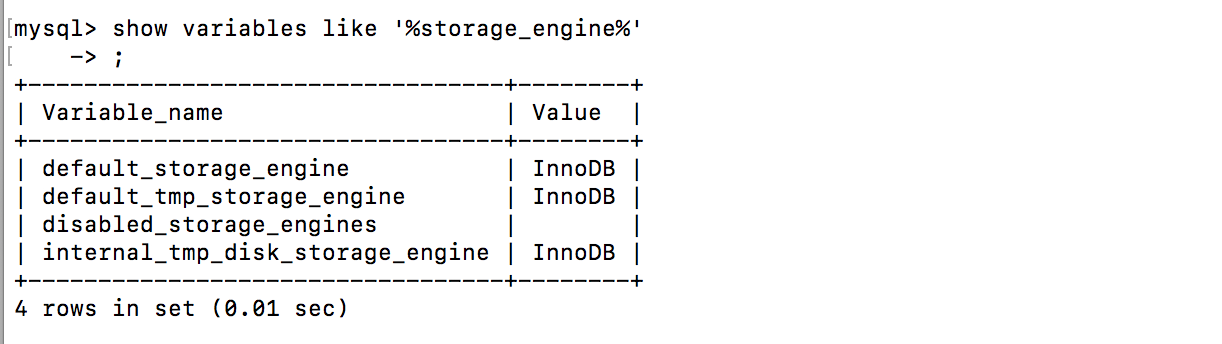
show engines





默认存储引擎

show variables like ‘%storage\_engine%’



指定数据库引擎：



## 2.3 索引

索引：相当于目录，

index是帮助数据库MYSQL高效获取数据的的数据结构（树：B树（默认），Hash树。。）

B树：小的放左边，大的放又边

索引弊端：

1. 索引本身需要占据存储空间，可以放在内存/硬盘中
2. 索引不是所有情况使用a》数据量不大，b》经常修改的列从c》很少使用的列
3. 索引提高查询效率，降低增删改的效率

优势：

1. 提高查询效率（降低了IO使用率）
2. 降低cpu的使用率，B树索引就是一个排好序的

B+树：数据全部存放在叶节点中，B+树种查询任意的数据次数N次

索引分类：

单值索引：单列，只有一列加了索引，一个表可以有多个单值索引

唯一索引：唯一索引不能重复，

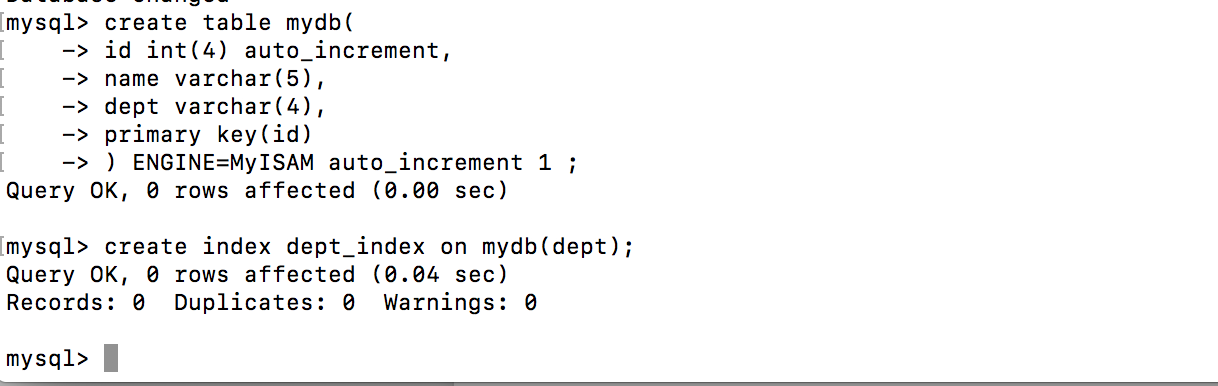
复合索引：多个列构成的索引（相当于二级目录），复合索引不一定两列都会用

方式一

创建索引：

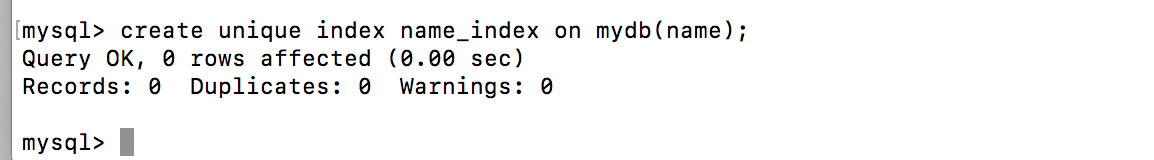
单值索引

create index dept\_index on mydb(dept);//单值索引



唯一索引:

create unique index name\_index on mydb(name);//唯一索引



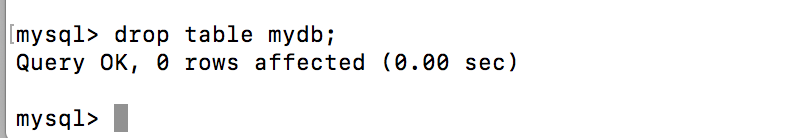
复合索引：

create index dept\_name\_index on mydb(dept,name);



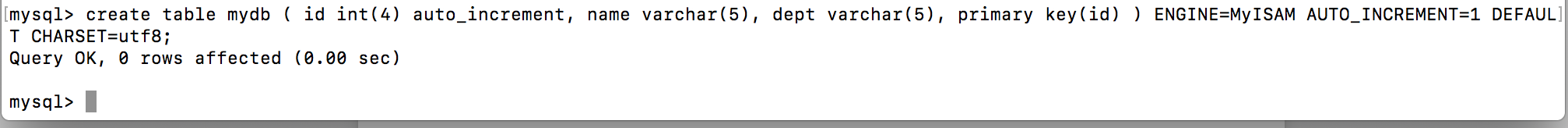
删除索引，删除表，

drop table mydb



重新创建表格

create table mydb ( id int(4) auto\_increment, name varchar(5), dept varchar(5), primary key(id) ) ENGINE=MyISAM AUTO\_INCREMENT=1 DEFAULT CHARSET=utf8;



方式二：

单值索引：

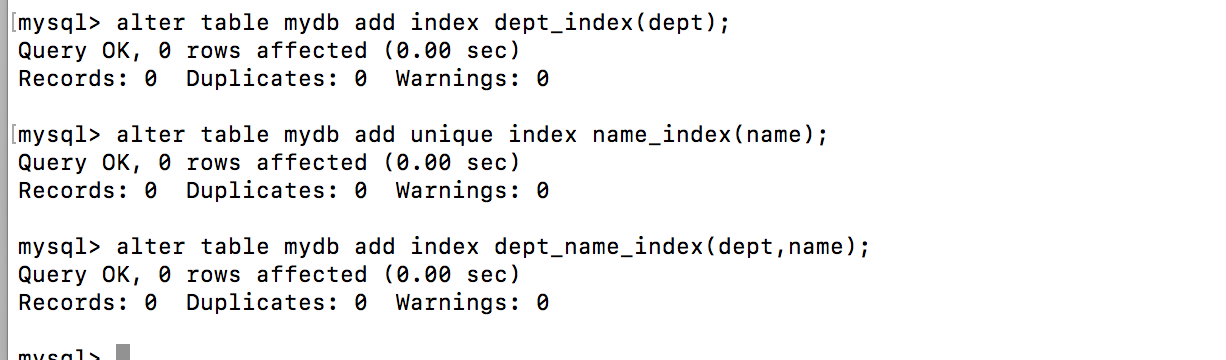
alter table mydb add index dept\_index(dept);

唯一索引：

alter table mydb add unique index name\_index(name);

复合索引：

alter table mydb add index dept\_name\_index(dept,name);



注意：如果一个字段是primary key,则该字段默认是主键索引；

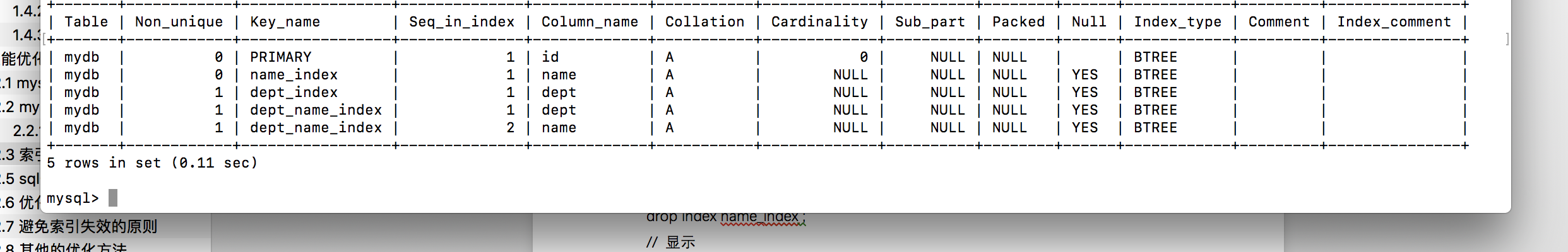
删除索引：

drop index 索引名 on 表名；

drop index name\_index on mydb ;

// 显示

show index from mydb;



## 2.5 sql性能问题,优化方法

问题产生原因：性能低，执行时间太长，等待时间太长，SQL语句欠佳（连接查询），索引失效，服务器参数设置不合理（缓冲区，线程数）

a.SQL语句：

编写过程：

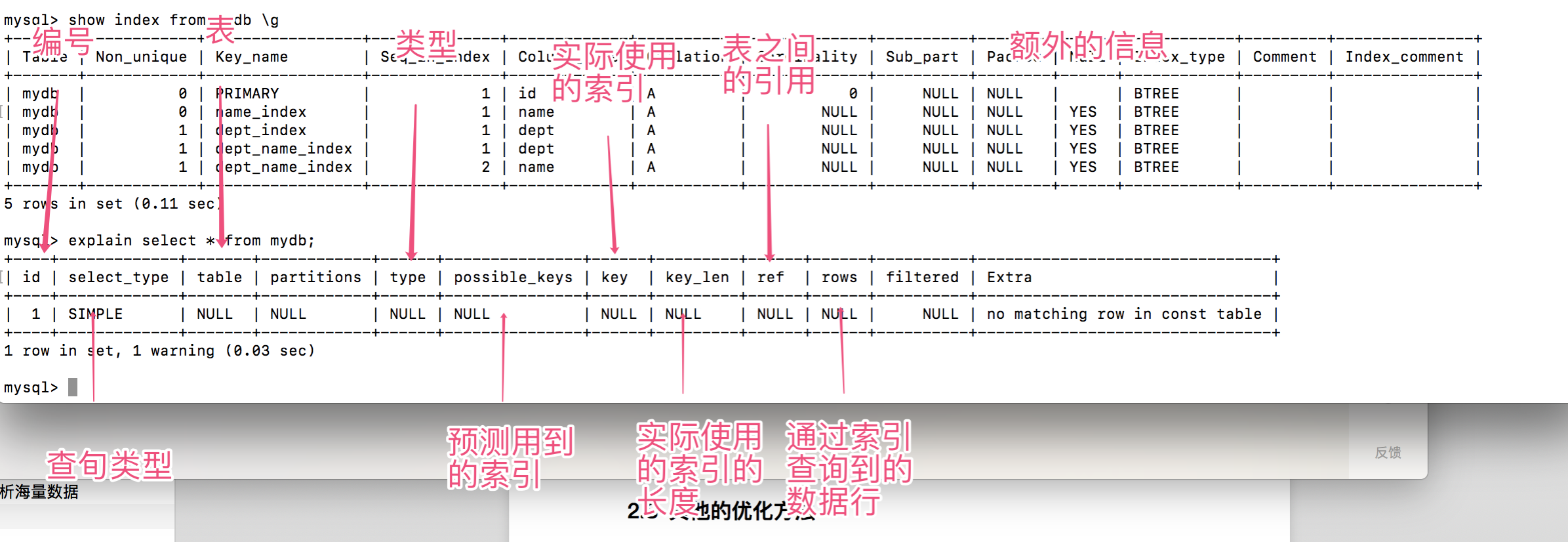
select ..from ….join ..on.. where ..group by ..having ..order by .. limit ..

解析过程：

from ..on.. join … where ..group by ..having..select dinstinct ..order by

b.SQL优化，主要是优化索引

1. 分析sql的执行计划：explain，可以模拟SQL优化器执行sql语句，
2. mysql查询优化会干扰我们的优化



创建3个表

create table course(

id int(3),

cname varchar(20),

tid int(3)

);

create table course(

id int(3),

cname varchar(20),

tid int(3)

);

create table teacher(

tid int(3),

tname varchar(20),

tcid int(3)

);

create table teacherCard(

tcid int(3),

tcdesc varchar(20)

);

insert into course values(1,'java',1);

insert into course values(2,'html',1);

insert into course values(3,'sql',2);

insert into course values(4,'web',3);

insert into teacher values(1,'tz',1);

insert into teacher values(2,'tw',2);

insert into teacher values(3,'tl',3);

insert into teacherCard values(1,'tzdesc');

insert into teacherCard values(2,'twdesc');

insert into teacherCard values(3,'tldesc');

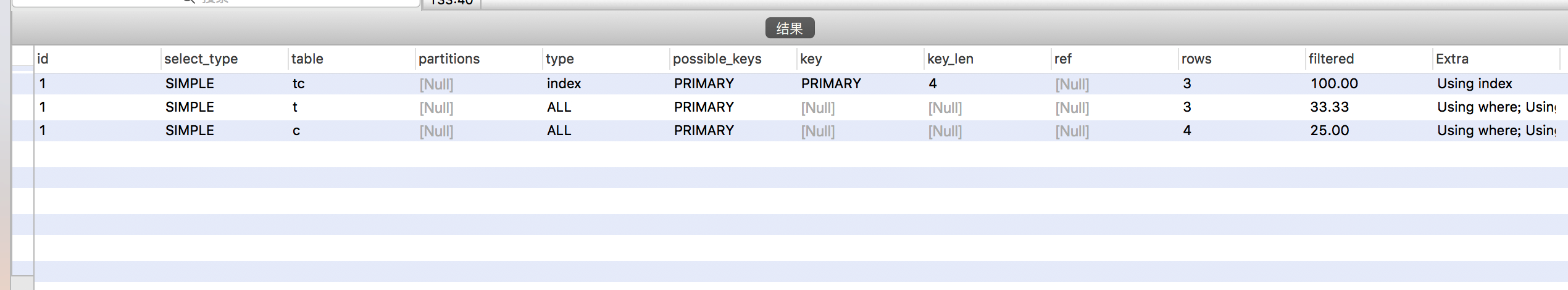
### 2.5.1 练习1（id）

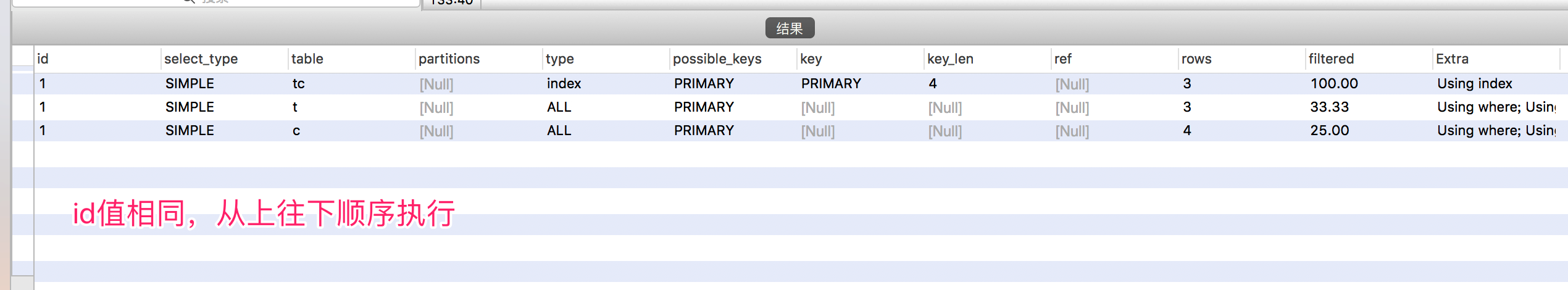
-- 查询课程编号为2 或 教师证编号为3 的老师信息

select t.\* from teacher t, course c, teacherCard tc where t.tid = c.tid and t.tcid = tc.tcid and (c.cid = 2 or tc.tcid = 3);

* explain

explain select t.\* from teacher t, course c, teacherCard tc where t.tid = c.tid and t.tcid = tc.tcid and (c.cid = 2 or tc.tcid = 3);





teacher加数据：

insert into teacher values(4,'t4',3);

delete from course where cid > 2;



数据量小，优先查询

### 2.5.2 练习2（id）

查询教授SQL课程的老师的描述

select tc.tcdesc from teacherCard tc ,course c,teacher t where t.tid = c.tid and t.tcid = tc.tcid and c.cname = 'sql' ;

explain select tc.tcdesc from teacherCard tc ,course c,teacher t where t.tid = c.tid and t.tcid = tc.tcid and c.cname = 'sql' ;



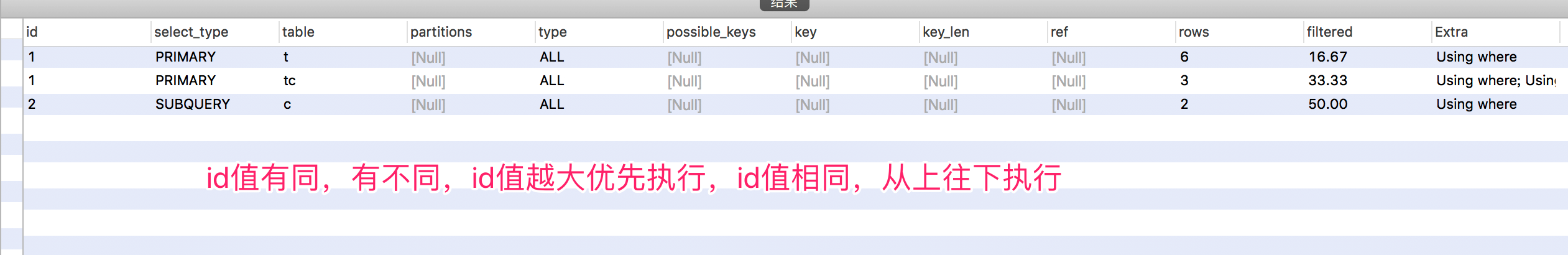
将上述的查询转成子查询

select tc.tcdesc from teacherCard tc where tcid = (select t.tcid from teacher t where t.tid = (select c.tid from course c where c.cname ='sql'))



子查询+多表

select t.tname ,tc.tcdesc from teacher t ,teacherCard tc where t.tcid = tc.tcid and t.tid = (select c.tid from course c where c.cname='sql' )



id值相同，从上往下执行，id值不同，优先执行id大的值；

### 2.5.3 select\_type

explain select tc.tcdesc from teacherCard tc where tcid = (select t.tcid from teacher t where t.tid = (select c.tid from course c where c.cname ='sql'))

PRIMARY:最外层一般是主查询

SUBQUERY：子查询，（非最外层）

SIMPLE：简单查询

DERIVED：衍生查询

#### 2.5.3.1 简单查询(simple)

explain select \* from teacher ;



#### 2.5.3.2 衍生查询(DERIVED：查询时用到临时表)

a．在from子查询中只有一张表

b．在from子查询中，如果有（table1 union table2，则table1就是衍生的）

explain select cr.cname from (select \* from course where tid = 1 union select \* from course where tid = 2) cr



union result也会出现衍生查询

## 2.6 优化案例

## 2.7 避免索引失效的原则

## 2.8 其他的优化方法

## 2.9 sql排查 ---慢查询日志

## 2.10 分析海量数据