**《数据库原理》实验报告**

**实验名称 数据查询分析实验**

**班 级 2014211302**

**学 号 2014211182**

**姓 名 曹桢**

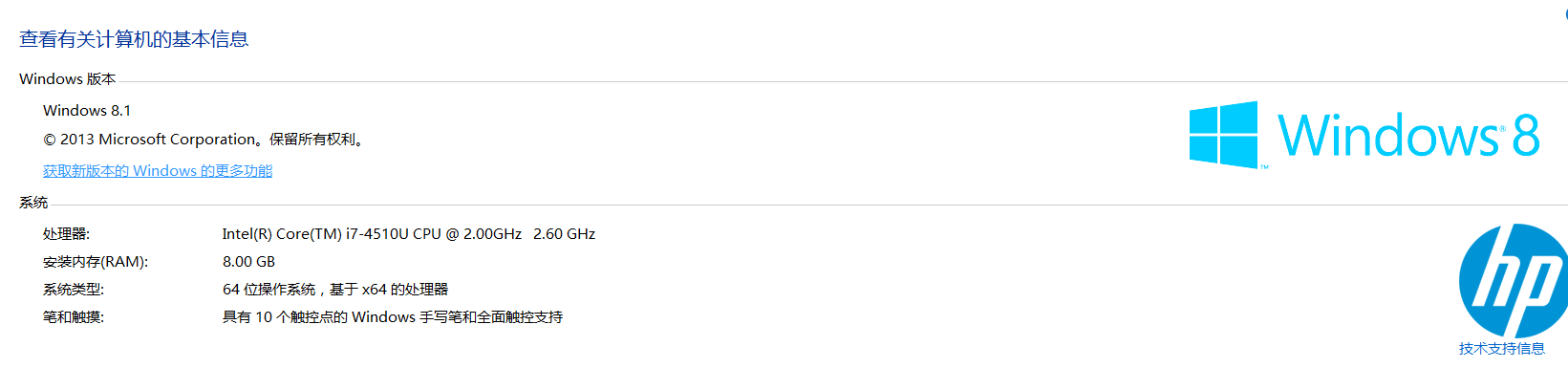
# 实验六 数据查询分析实验

### 一、实验目的

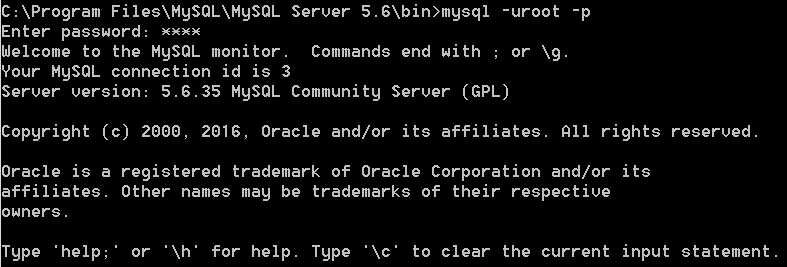
通过对不同情况下查询语句的执行分析，巩固和加深对查询和查询优化相关理论知识的理解，提高优化数据库系统的实践能力，熟悉了解mysql中查询分析器的使用，并进一步提高编写复杂查询的SQL 程序的能力。

**二、实验平台及环境**

* 操作系统环境：windows8.1 64位



* 数据库版本：mysql 5.6



**三、实验内容**

1.索引对查询的影响

1）对结果集只有一个元组的查询分三种情况进行执行（比如查询一个具体学生的信息）：不建立索引，（学号上）建立非聚集索引，（学号上）建立聚集索引。

用查询分析器的执行步骤和结果对执行进行分析比较。

2）对结果集中有多个元组的查询（例如查看某门成绩的成绩表）分类似（1）的三种情况进行执行比较。

3）对查询条件为一个连续的范围的查询（例如查看学号在某个范围内的学生的选课情况）分类似（1）的三种情况进行执行比较，注意系统处理的选择。

4）索引代价。在有索引和无索引的情况下插入数据（例如在选课情况表SC 上插入数据），比较插入的执行效率。

2.对相同查询功能不同查询语句的执行比较分析

1)group by

select avg(grade)

from sc

group by cno

having cno =100

select avg(grade)

from sc

where cno = 100

有和没有group by，比较其查询效率，并分析。

2)

select sno，sname，bdate

from student s1

where bdate =

(select max(bdate)

from student s2

where s1.dept = s2.dept

)

另一个：

select dept ，max(bdate) as maxAge into tmp

from student

group by dept;

select sno, sname , bdate

from student，tmp

where student. bdate = tmp.maxAge and tmp.dept=student.dept

drop table tmp;

重写后的查询一定比原始查询更优吗？通过执行分析结果。

3)对下面两个查询进行比较

select sname，bdate

from student

where dept != 10 and bdate > all

(select bdate

from student

where dept = 10

)

另：

select sname , bdate

from student

where dept != 10 and bdate >

( select max(bdate)

from student

where dept = 10

)

3.查询优化

除了建立适当索引，对**SQL** 语句重写外，还有其他手段来进行查询调优，例如调整缓冲区大小，事先建立视图等。设计实现下列查询，使之运行效率最高。

写出你的查询形式，以及调优过程；并说明最优情况下的运行时间。

1)查找选修了每一门课的学生。

2)查找至少选修了课程数据库原理和操作系统的学生的学号。

### 四、实验步骤和结果分析

#### 1.索引对查询的影响

**1）对结果集只有一个元组的查询分三种情况进行执行**

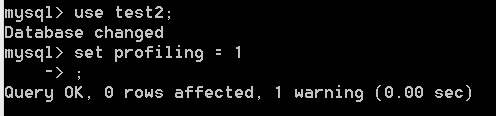
set profiling = 1;

use test2;

select \* from student where sno = '31402';

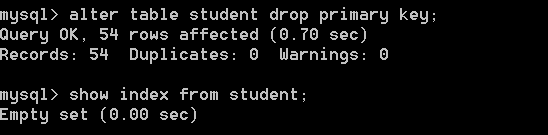
show profiles;

show profile for query n;

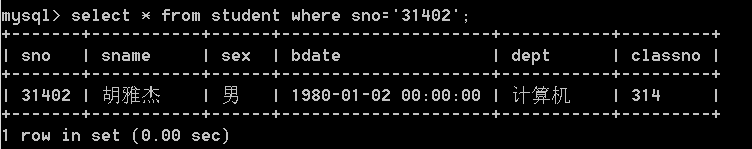


**a.不建立索引**

删除主索引：



执行select \* from student where sno = '31402';

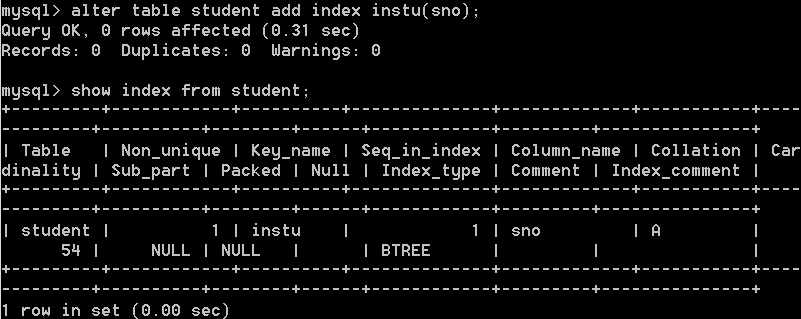


查看效率Show profiles; 

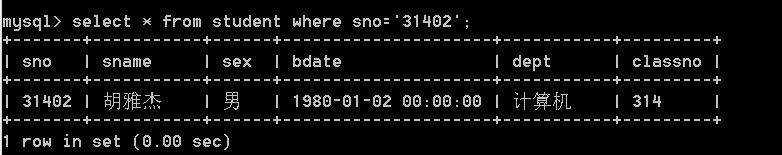
执行0.00025475s

**b.学号上建立非聚集索引**

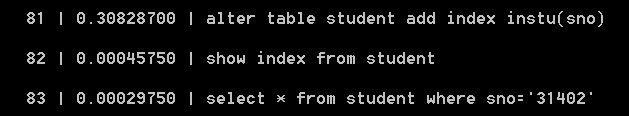
建立索引instu



执行命令



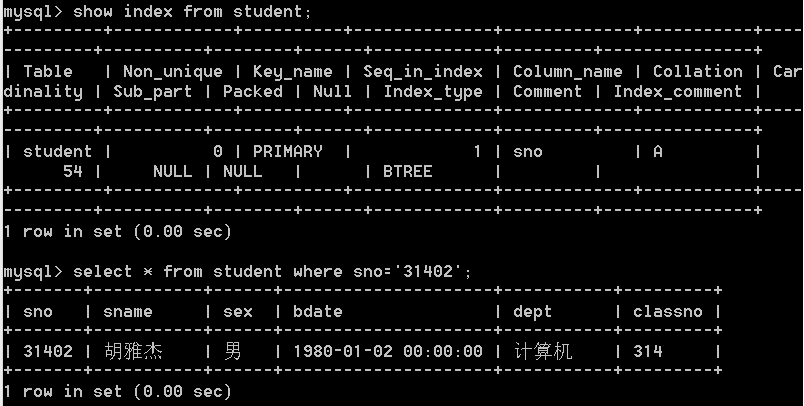
查看效率



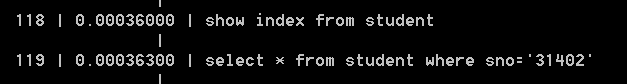
执行0.00029750s

**c.学号上建立聚集索引**

查看索引并执行命令



查看效率



执行时间为0.00036300s

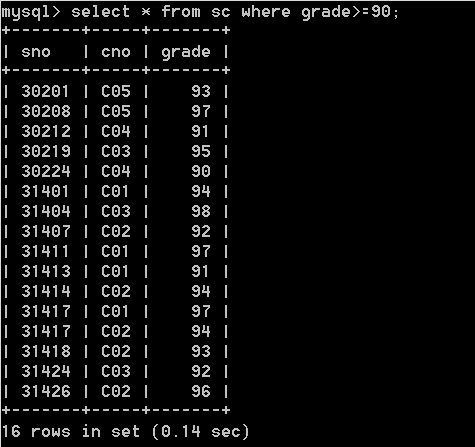
比较：

无索引、非聚集索引和聚集索引的时间分别是0.00025475s、0.00029750s、0.00036300s。执行时间相差不大，都在同一个数量级下，所以结果集只有一个元组的查询，三种情况耗时相当。

**2）对结果集中有多个元组的查询（例如查看某门成绩的成绩表）分类似（1）的三种情况进行执行比较。**

**A、无索引**

执行指令

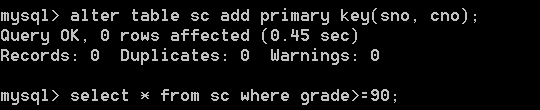


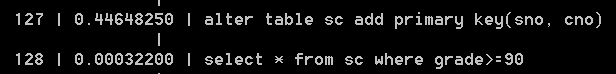
查看效率



耗时0.00102875s

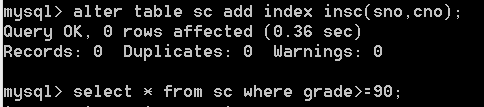
**B、聚集索引**





耗时0.00032200s

**C、非聚集索引**





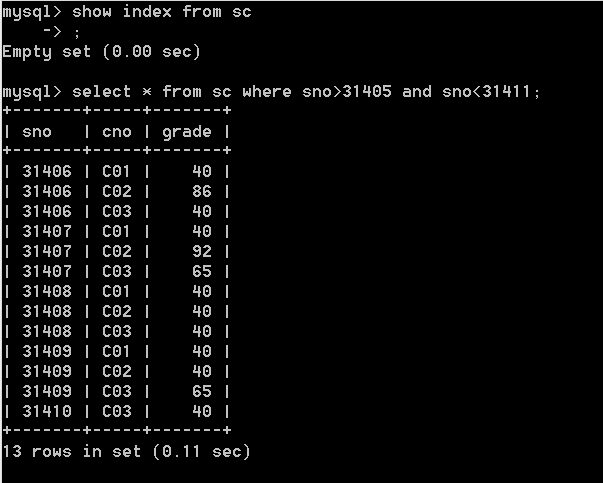
耗时0.00032100s

**比较：**

无索引耗时0.00102875s，聚集索引耗时0.00032200s，非聚集索引耗时0.00032100s。所以结果集中有多个元组的查询，有聚集或非聚集索引比无索引要快，而聚集和非聚集索引的耗时差不多。

**3）对查询条件为一个连续的范围的查询（例如查看学号在某个范围内的学生的选课情况）分类似（1）的三种情况进行执行比较，注意系统处理的选择。**

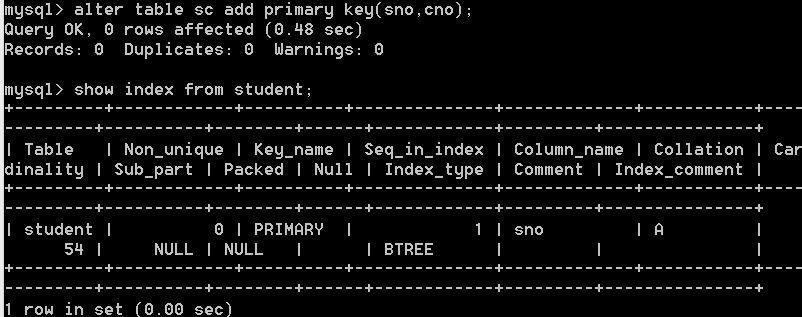
A、无索引

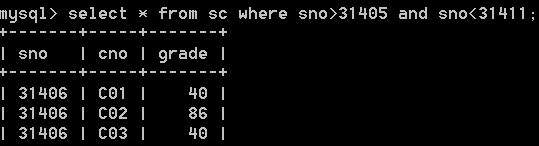




耗时0.10734475s

B、聚集索引

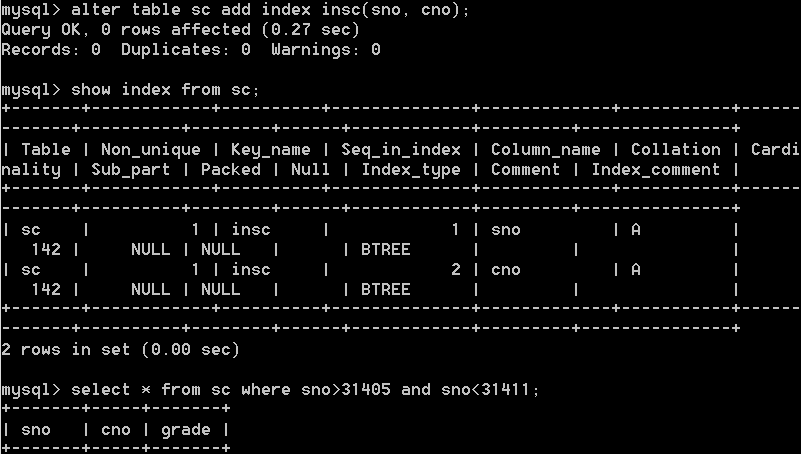


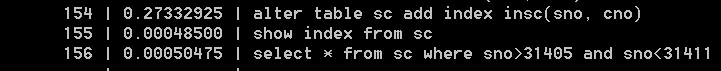




耗时0.00033875s

C、非聚集索引





耗时0.00050475s

**比较：**

无索引耗时0.10734475s，聚集索引耗时0.00033875s，非聚集索引耗时0.00050475s。由此可见，查询条件为一个连续的范围的查询时，查询速度：聚集索引>非聚集索引 > 无索引。

**4）索引代价。在有索引和无索引的情况下插入数据（例如在选课情况表SC 上插入数据），比较插入的执行效率。**

insert into sc values(‘31430’,’C01’,’100’);

insert into sc values(‘31430’,’C02’,’ 100’);

insert into sc values(‘31430’,’C03’,’ 100’);

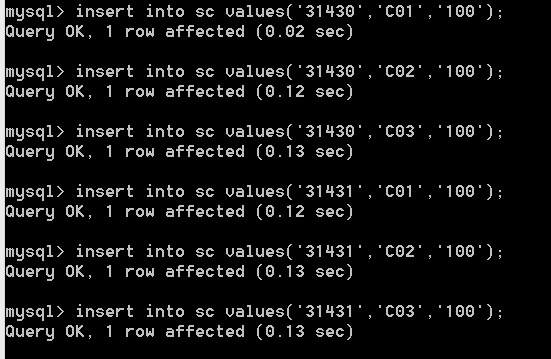
insert into sc values(‘31431’,’C01’,’30’);

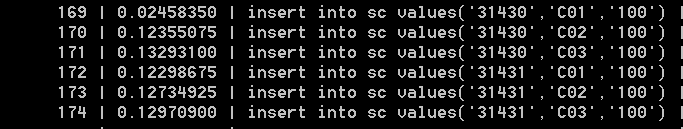
insert into sc values(‘31431’,’C02’,’30’);

insert into sc values(‘31431’,’C03’,’30’);

a、无索引

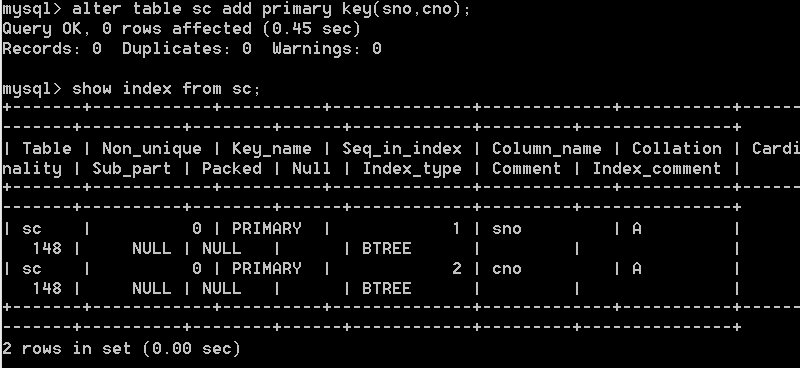


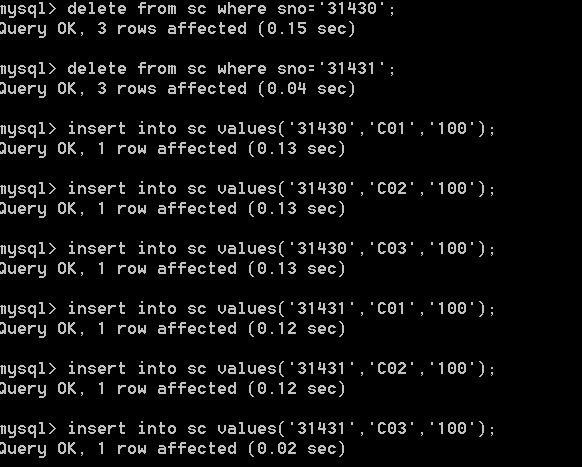


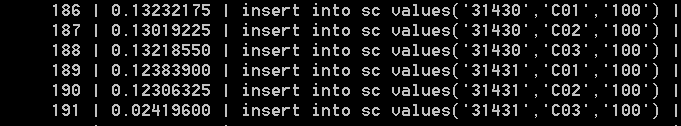


耗时共0.66111025s

b、聚集索引

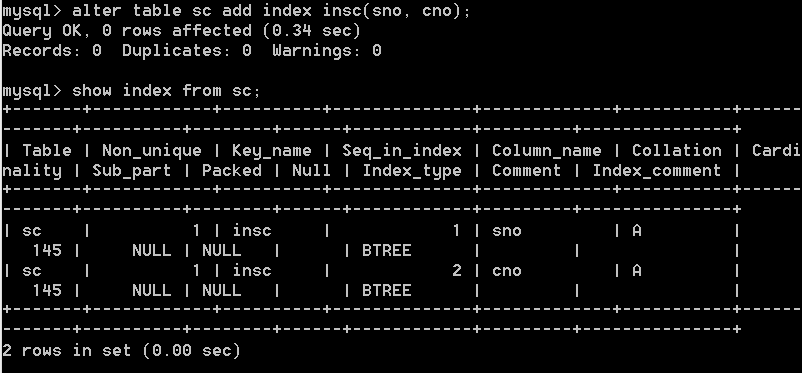


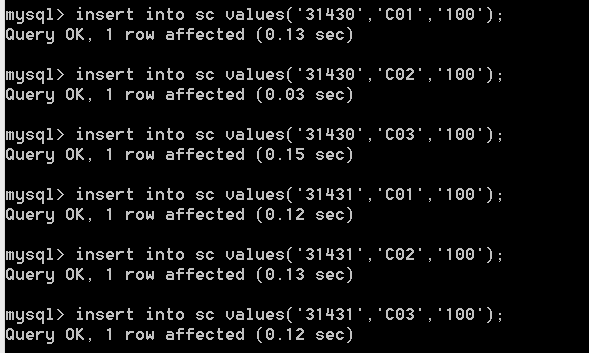


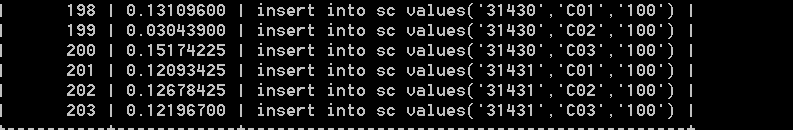


耗时共0.66599775s

c、非聚集索引







共耗时0.68296275s

**比较：**

对于插入6条数据，无索引耗时0.66111025，聚集索引耗时0.66599775，非聚集索引耗时0.68296275s。三者相差不大，这说明插入数据，又索引并不能优化性能。

#### 2.对相同查询功能不同查询语句的执行比较分析

1)group by

(1)select avg(grade)

from sc

group by cno

having cno =100;

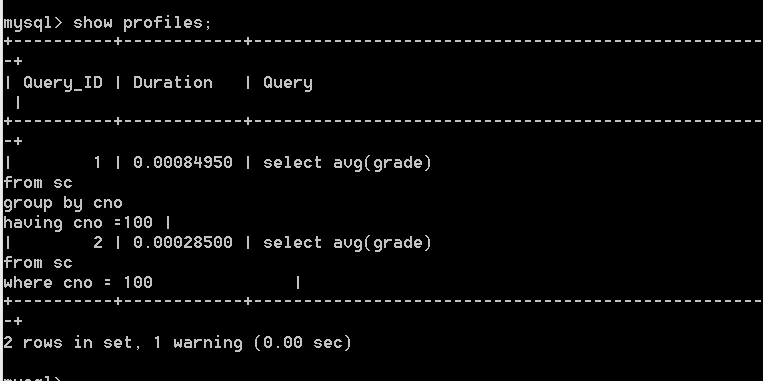
(2)select avg(grade)

from sc

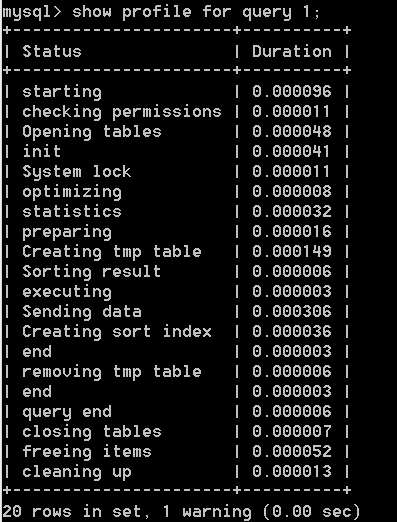
where cno = 100;

结果：

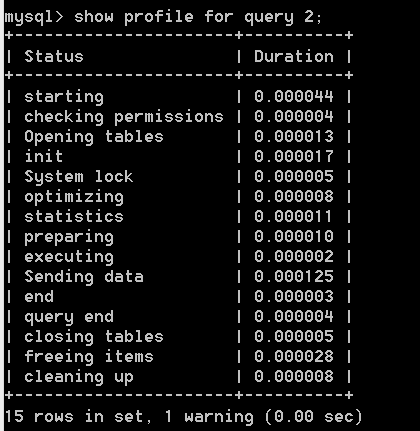
show profiles;



show profile for query 1;



show profile for query 2;



有group by的情况，耗时0. 00084950s；没有group by的情况，耗时0.00028500s。没有group by的情况更快一些。因为有group by比之没有的情况，多了create tmp table，sorting result，create sort index这几步，并且在starting，checking permissions，Opening tables，init等几步，有group by的情况也耗费更多一点的时间。

2)重写

(1)select sno, sname, bdate

from student s1

where bdate =

(select max(bdate)

from student s2

where s1.dept = s2.dept

);

(2) create table tmp as

(select dept, max(bdate) as maxBdate

from student

group by dept);

select sno, sname, bdate

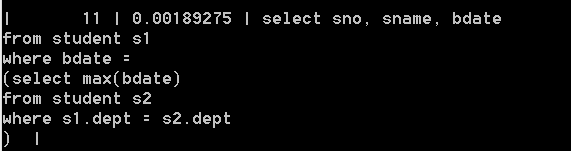
from student, tmp

where student.bdate=tmp.maxBdate and tmp.dept=student.dept;

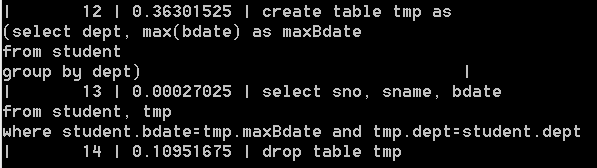
drop table tmp;

**结果：**

(1) 原始查询：



(2)重写后：



**分析：**

原始查询，耗时0.00189275s；重写后，查询耗时0.00027025s。重写后查询时间明显下降，相当于时间减小了一个数量级。

但是，重写后，建表耗时0.36301525s，删表耗时0.10951675s，导致重写后查询的总时间为0.47280225s，远远大于原始查询时间。

由此可见，重写可以缩短查询的时间，但是建标和删表会造成**较大的开销**。因此，只有在建表后，会很多次查询的时候（多到可以建表和删表的开销不足为惧），重写后的查询才会比原始的更优。

3)对下面两个查询进行比较

(1)

select sname, bdate from student

where dept != '计算机' and bdate>all

(select bdate from student

where dept = '计算机');

(2)

select sname , bdate from student

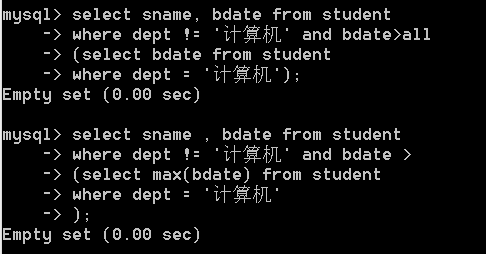
where dept != '计算机' and bdate >

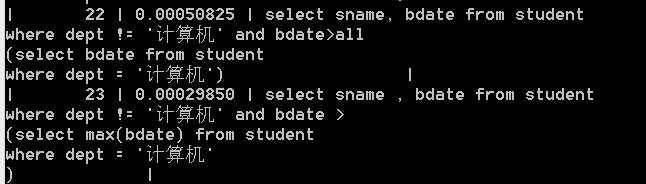
(select max(bdate) from student

where dept = '计算机'

);

**结果：**





**分析：**

两个查询语句从意义上讲等价，查询的结果集均为空。(1)耗时0.00050825s，(2)耗时0.00029850s。第二个语句耗时比第一个语句更少，效率更高。

#### 3.查询优化

除了建立适当索引，对**SQL** 语句重写外，还有其他手段来进行查询调优，例如调整缓冲区大小，事先建立视图等。设计实现下列查询，使之运行效率最高。

写出你的查询形式，以及调优过程；并说明最优情况下的运行时间。

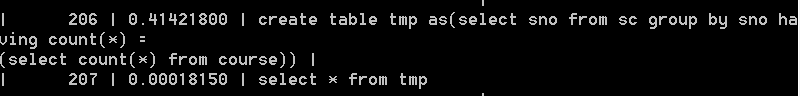
1)查找选修了每一门课的学生。

A、事先创建表

create table tmp as(select sno from sc group by sno

having count(\*)=(select count(\*) from course));

select \* from tmp;



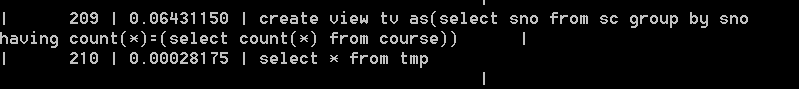
建表耗时0.41421800s，查询耗时0.00018150s。共耗时0.41439950s

B、事先创建视图

create view tv as(select sno from sc group by sno

having count(\*)=(select count(\*) from course));

select \* from tmp;



建视图耗时0.06431150s，查询耗时0.00028175s。共耗时0.06459325s

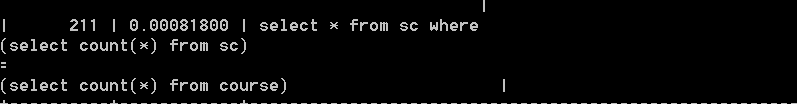
C、嵌套子查询

select \* from sc where

(select count(\*) from sc)

=

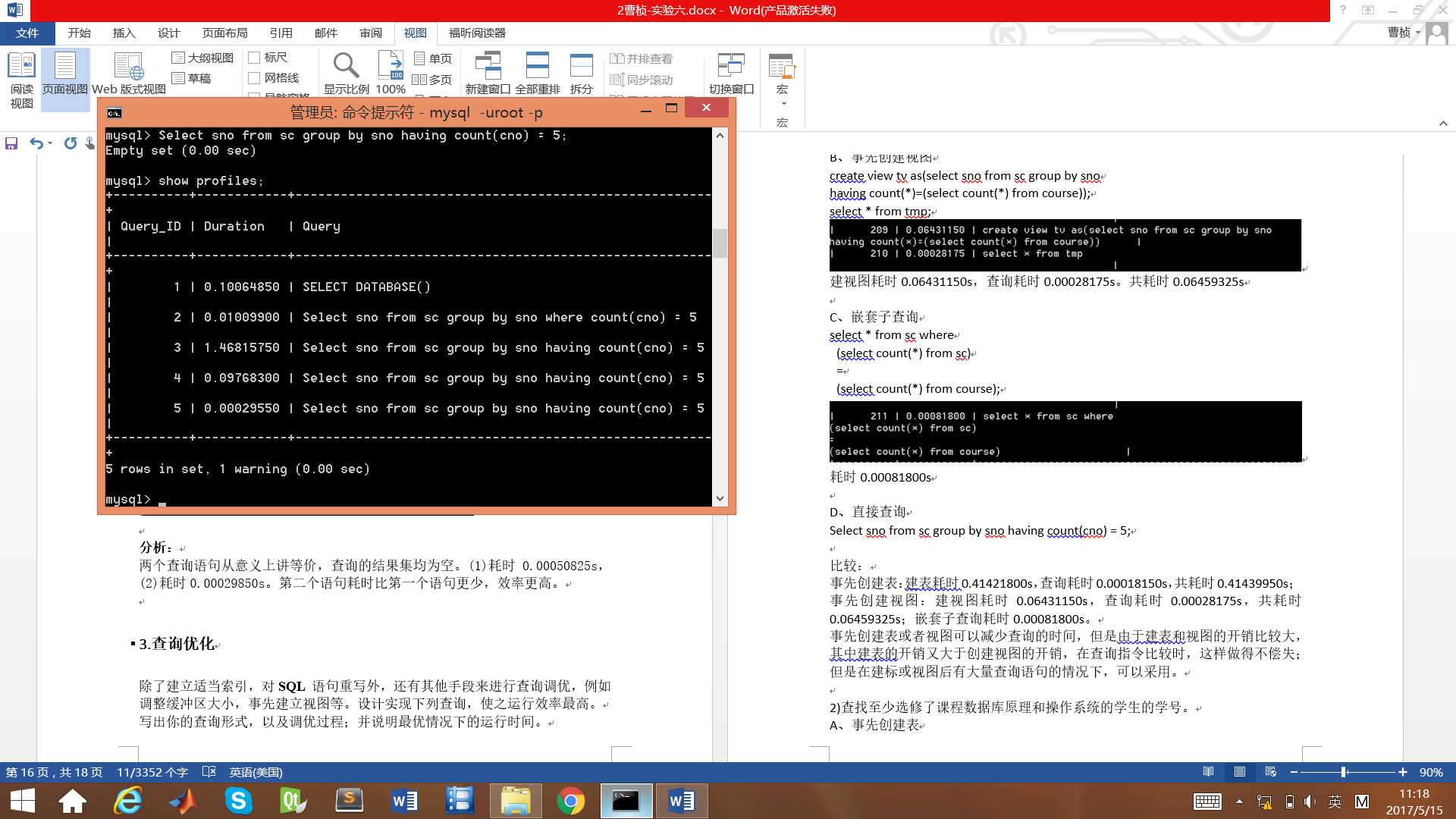
(select count(\*) from course);



耗时0.00081800s

D、直接查询

Select sno from sc group by sno having count(cno) = 5;



耗时0.0002s

比较：

事先创建表：建表耗时0.41421800s，查询耗时0.00018150s，共耗时0.41439950s；事先创建视图：建视图耗时0.06431150s，查询耗时0.00028175s，共耗时0.06459325s；嵌套子查询耗时0.00081800s。

事先创建表或者视图可以减少查询的时间，但是由于建表和视图的开销比较大，其中建表的开销又大于创建视图的开销，在查询指令比较时，这样做得不偿失；但是在建标或视图后有大量查询语句的情况下，可以采用。

2)查找至少选修了课程数据库原理和操作系统的学生的学号。

A、事先创建表

create table t as

(select sc.sno as sno, cname from sc, course

where sc.cno=course.cno and

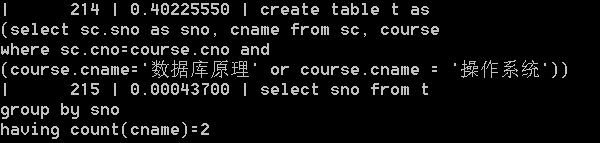
(course.cname='数据库原理' or course.cname = '操作系统'));

查询：

select sno from t

group by sno

having count(cname)=2;



创建表耗时0.40225550s，查询耗时0.00043700s，共耗时0.40269250s

B、事先创建视图

create view ttv as

(select sc.sno as sno, cname from sc, course

where sc.cno=course.cno and

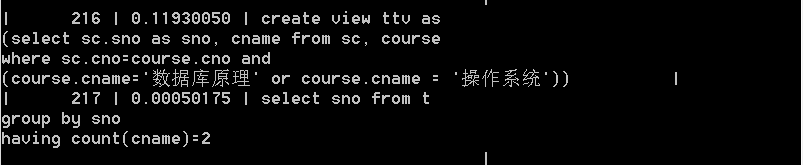
(course.cname='数据库原理' or course.cname = '操作系统'));

查询：

select sno from t

group by sno

having count(cname)=2;



创建视图耗时0.11930050s，查询耗时0.00050175s，共耗时0.11980225s

比较：

事先创建表：建表耗时0.40225550s，查询耗时0.00043700s，共耗时0.40269250s；事先创建视图：建视图耗时0.11930050s，查询耗时0.00050175s，共耗时0.11980225s。

建表的开销比建立视图更大，因此，在需要事先创建表或视图来查询时，最好创建视图。

**五、实验结果分析**

1、在结果集中有一个元组的时候，聚集索引、非聚集索引和没有索引性能上类似；在结果集中有多个元组的时候，聚集索引、非聚集索引性能相似，都优于没有索引；在查询条件为一个连续范围时聚集索引优于非聚集索引优于没有索引；在插入一些元组时，聚集索引、非聚集索引和没有索引性能上类似。

2、对于要得到某一结果集，可以事先创建表、事先创建视图、嵌套子查询。要事先创建表或视图，有额外开销，创建表的开销大于创建视图的开销，但是查询时间两者优于嵌套子查询。在创建表或视图后又多次查询，足够抵消开销时，可以采用事先创建表或者视图的办法，但是如果只有少量查询，比如一次，则采用嵌套子查询更优。

**六、实验小结**

(一)问题和收获

1、在最初实验的时候，不知道聚集索引和非聚集索引是什么意思，他们的sql语句是什么样子。通过查询资料，得知在mysql中，聚集索引是主键索引，在设置主键的时候就已经默认设置为主索引；非聚集索引就是mysql的普通索引。聚集索引：该索引中键值的逻辑顺序决定了表中相应行的物理顺序；非聚集索引：数据存储在一个地方，索引存储在另一个地方，索引带有指针指向数据的存储位置。

2、实验开始，不知道如何查看查询的性能，因为在课程学习中尚未学到。通过查询资料，知道通过

set profiling=1

show profiles

可以查看数据库中各种操作的执行时间。

(二)心得

通过这次的数据库实验，我了解了数据库聚集索引和非聚集索引的原理、sql语句和适用情景。学会了在同一个目的有多种mysql语句实现方法时，如何比较这些实现方法的性能。这些都是在课堂学习中没有学到的知识，但是确实很实用的知识，让我收获颇多，受益匪浅。