

数据查询分析实验

【实验六】



目录

[一． 数据查询分析实验 1](#_Toc514690426)

[1． 实验目的 1](#_Toc514690427)

[2． 实验环境 1](#_Toc514690428)

[3． 实验内容与要求 1](#_Toc514690429)

[4． 实验步骤及结果分析 3](#_Toc514690430)

[5． 实验小结 14](#_Toc514690431)

2018-5-18

[裴子祥 计科七班 学号2015211921]

[指导老师：杜军平]

# 数据查询分析实验

1. 实验目的

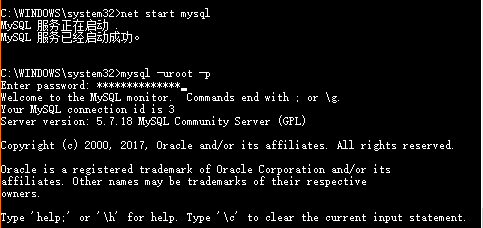
通过对不同情况下查询语句的执行分析，巩固和加深对查询和查询优化相关理论知识的理解，提高优化数据库系统的实践能力，熟悉了解mysql中查询分析器的使用，并进一步提高编写复杂查询的SQL 程序的能力。

1. 实验环境

**Microsoft Windows 10 专业版 64位**



**数据库版本：5.7.18 MySQL Community Server (GPL)**



1. 实验内容与要求
2. **索引对查询的影响**
3. 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

)

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）

select sname，bdate

from student

where dept != 10 and bdate > all

(select bdate

from student

where dept = 10

)

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。
3. 实验步骤及结果分析

**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;



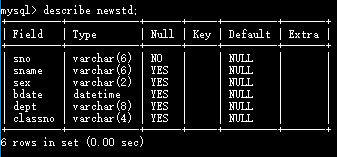
select \* from std where sno = '31402';

show profiles;

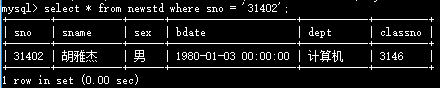
show profile from query n;//查询第n条sql语句执行时间详情

1. **不建立索引**

表newstd，与std内容相同，无主键索引。



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



show profiles;查看效率



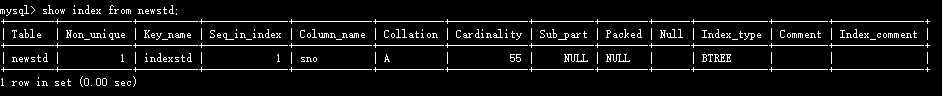
执行时间0.0004s

1. **学号建立非聚集索引**

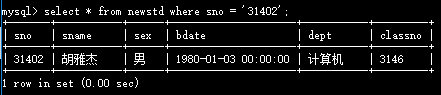
alter table newstd add index indexstd(sno);



show index from newstd;



执行查询操作



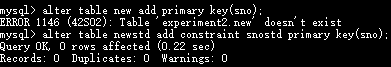
查看效率



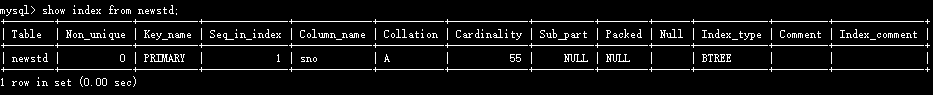
执行时间0.0027s

1. **学号建立聚集索引**

添加学号为主键约束



显示索引



执行并查看效率



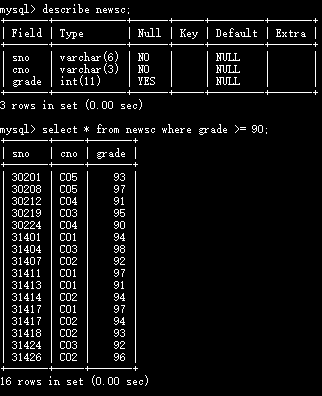
执行时间为0.00022s

**比较结果：**

无索引、非聚集索引和聚集索引的时间分别是0.0004s、0.0027s、0.00022s。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

1. **不建立索引**



查看效率，执行时间0.00035325s



1. **非聚集索引**

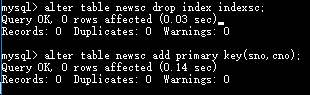


执行并查看效率，执行时间0.000285s



1. **聚集索引**

删除非聚集索引并创建聚集索引



执行并查看效率，执行时间0.00029975s

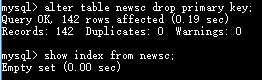


**比较结果：**

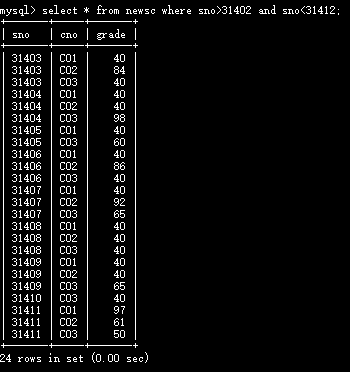
无索引耗时0.00035325s，聚集索引耗时0.00029975s，非聚集索引耗时0.000285s。所以结果集中有多个元组的查询，有聚集或非聚集索引比无索引要快，而聚集和非聚集索引的耗时差不多。

（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。

1. **不建立索引**



select \* from newsc where sno>31402 and sno<31412;



查看效率，执行时间0.00027825s



1. **非聚集索引**



执行并查看效率，执行时间0.00041325s



1. **聚集索引**

删除非聚集索引并创建聚集索引

执行并查看效率，执行时间0.00028600 s



**比较结果：**

无索引耗时0.00027825s，聚集索引耗时0.00028600 s，非聚集索引耗时0.00041325s。由此可见，查询条件为一个连续的范围的查询时，查询速度：无索引>聚集索引>非聚集索引。

（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 newsc values(‘31430’,’C01’, 100);

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

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

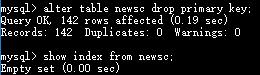
insert into newsc values(‘31431’,’C01’,30);

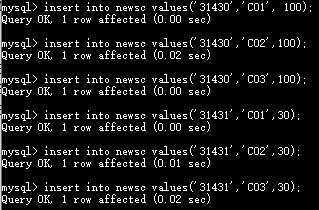
insert into newsc values(‘31431’,’C02’,30);

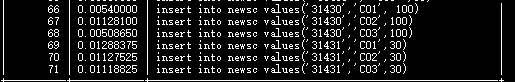
insert into newsc values(‘31431’,’C03’,30);

1. **不建立索引**

删除聚集索引



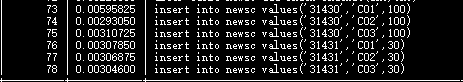




查询效率，执行时间

1. **非聚集索引**





执行并查看效率，执行时间

1. **聚集索引**

删除非聚集索引并创建聚集索引



三者相差不大，好像无索引最慢

**2、对相同查询功能不同查询语句的执行比较分析**

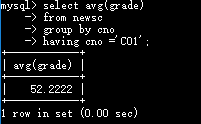
（1）group by

select avg(grade)

from newsc

group by cno

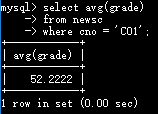
having cno =’C01’;

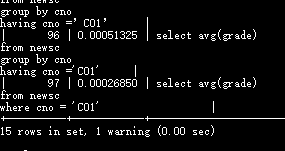


select avg(grade)

from newsc

where cno = 'C01';





有group by的情况，耗时0.00051325；没有group by的情况，耗时0.00026850。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

原始：

select sno,sname,bdate

from newstd s1

where bdate =

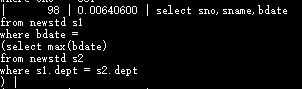
(select max(bdate)

from newstd s2

where s1.dept = s2.dept

);





查询效率，执行时间0.06404s

重写：

create table tmp as

(select dept,max(bdate) as maxBdate

from newstd

group by dept);



建表时间,0.13743350s

select sno,sname,bdate

from newstd,tmp

where newsstd.bdate = tmp.maxBdate and tmp.dept = newstd.dept;



重写后查询时间0.0009895s

drop table tmp;



删表时间0.01593375s

**比较结果：**

原始查询，耗时0.06404s；重写后，查询耗时0.0009895s。重写后查询时间明显下降。

但是，重写后，建表耗时0.13743350s，删表耗时0.01593375s，导致重写后查询的总时间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，但是建表和删表会造成较大的开销。因此，只有在建表后，如果有多次查询的时候（多到建表和删表的可以开销忽略），重写后的查询才会比原始的更好。

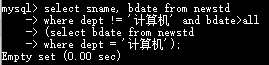
（3）对下面两个查询进行比较

select sname, bdate from newstd

where dept != '计算机' and bdate>all

(select bdate from newstd

where dept = '计算机');

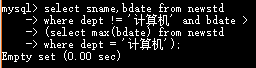


select sname,bdate from newstd

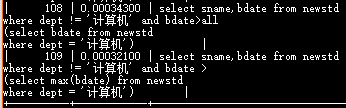
where dept != '计算机' and bdate >

(select max(bdate) from newstd

where dept = '计算机');



查询效率:



**比较结果：**

两个查询语句查询的结果集均为空。前者耗时0.00034300s，后者耗时0.00032100s。两个句子耗时相当。

**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. 事先创建表

create table tmp as

(select sno from newsc

group by sno

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



建表耗时，0.07801025s

select \* from tmp;



查询耗时，0.00016800s

总耗时0.07817825s

1. 事先创建视图

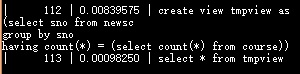
create view tmpview as

(select sno from newsc

group by sno

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

select \* from tmpview;



建视图耗时0.00839575s，查询耗时0.00098250s，总耗时，0.00937825s

1. 嵌套子查询

select \* from newsc where

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



查询耗时，0.00175900s

1. 直接查询

select sno from newsc group by sno having count(cno) = 5;



查询耗时，0.00051775s

**比较结果：**

就查询来看，事先创建表>直接查询>事先建立视图>嵌套子查询。但这可能和当时电脑使用情况有关，第二次同样查询视图时，可能时间比直接查询更少，前三者差距不大。对于建表和建视图，建表的创建耗时比建视图大，当面临这两个选择时，可以选择创建视图。当查询语句比较少的时候还是使用直接查询或嵌套子查询为妙。

1. 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。
2. 事先创建表

先找出选修了其中一门课的所有学生，再找出选出选项了两们的学生。

create table tmp2 as

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

where newsc.cno = course.cno and

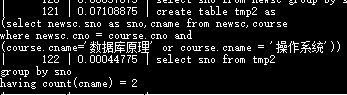
(course.cname='数据库原理' or course.cname = '操作系统'));

select sno from tmp2

group by sno

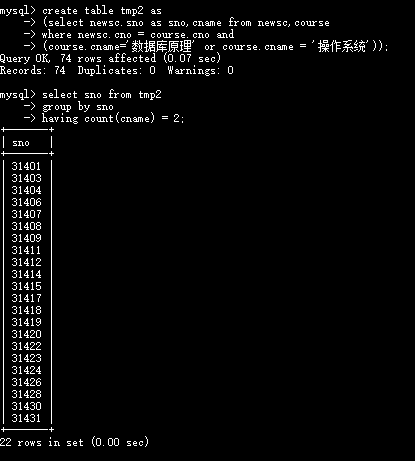
having count(cname) = 2;

查看效率



创建表时间0.07108875s，查询时间0.00044775s

查询结果如下：



1. 事先创建视图

create view tmpview2 as

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

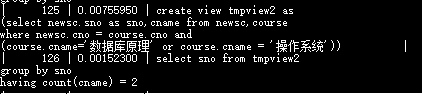
where newsc.cno = course.cno and

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

select sno from tmpview2

group by sno

having count(cname) = 2;



查询效率，创建视图时间0.00755950s，查询时间0.00152300s

1. 先查询选了’数据库原理’的学生,再查询其中选了’操作系统’的学生

select sno

from newsc natural join course

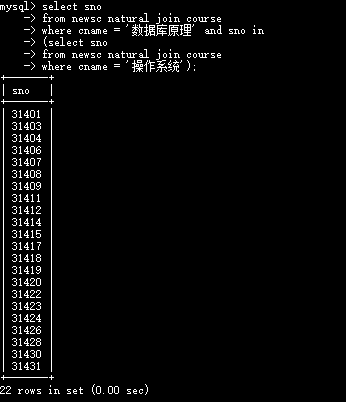
where cname = '数据库原理' and sno in

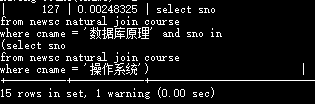
(select sno

from newsc natural join course

where cname = '操作系统');

查询结果：





查询耗时0.00248325s

比较结果：

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。

前两种是找出选择了其中任意一门课的学生，再选出选了两门课的学生。

逻辑上，第三种方式中，先找出一门课学生，再找其中选修另一门课学生，这样的优化是最快的。

1. 实验小结

**实验结果：**

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；
2. 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；
3. 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；
4. 在插入一些元组时，聚集索引、非聚集索引差不多，两者优于没有索引性能上类似
5. 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。
6. 查询优化中，可以先找出满足部分条件的结果，再去找满足下一个条件的结果，这样依次分层查找向下细化，并每次使用映射至找出需要的属性，这样可以精简数据库大小与复杂性，提升查询速度。

**问题记录：**

1. 不知聚集索引与非聚集索引的含义。查询资料得知，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. 不知如何查询sql语句的执行效率。通过询问同学得知：

set profiling=1;

show profiles;

show profile from query n;

可以查看，语句的执行效率（时间）。

**心得总结：**

这次实验，是所有实验以来费事最长的实验，主要在于各种查询方式的比较。学会了获得同一结果集的各种不同的sql语句组长表达方式，并通过比较性能，得知在不同场景下，需要如何选择适当的查询语句以达到最优性能。实践中，对sql语句的掌握更加熟练，收获满满。