

北京邮电大学数据库系统实验报告

实验名称: 数据查询分析实验

计算机科学与技术系 2015211306班

魏晓 学号:2015211301

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 Server 5.5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。
-

实验要求

- 用SQL语句完成以上操作
 - 要求学生独立完成以上内容。
 - 实验完成后完成要求的实验报告内容。
-

实验平台及环境

实验平台:MySQL 14.14

运行环境:Mac OS High Sierra 10.13(17A405)

可视界面:MySQL WorkBench 6.3.10

```
Xiaos-MacBook-Pro:~ weixiao$ mysql --version
```

```
mysql Ver 14.14 Distrib 5.7.20, for macos10.12 (x86_64) using EditLine wrapper
```

实验内容

索引对查询的影响

- MySQL没有聚集索引和非聚集索引之分,在这里直接给出结果
- 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），用查询分析器的执行步骤和结果对执行进行分析比较。
 - 不建立索引，在student表中查询学号为“30203”的学生
 - 由图可以看出执行时间大约为6毫秒

```
[mysql> select * from student where sno='30203';
```

| sno | sname | sex | birthdate | dept | classno | nation |
|-------|-------|-----|---------------------|------|---------|--------|
| 30203 | 唐雷 | 男 | 1980-01-02 00:00:00 | 电信 | 302 | NULL |

1 row in set (0.00 sec)

```
[mysql> show profiles;
```

| Query_ID | Duration | Query |
|----------|------------|---|
| 1 | 0.00009400 | SELECT DATABASE() |
| 2 | 0.00033900 | show databases |
| 3 | 0.00044400 | show tables |
| 4 | 0.00609100 | select * from student where snp='30203' |
| 5 | 0.00023500 | select * from student where sno='30203' |

5 rows in set, 1 warning (0.01 sec)

```
mysql> █
```

- 详细的执行步骤为

```
mysql> show profile for query 2;
```

| Status | Duration |
|----------------------|----------|
| starting | 0.000024 |
| checking permissions | 0.000005 |
| Opening tables | 0.000032 |
| init | 0.000007 |
| System lock | 0.000004 |
| optimizing | 0.000003 |
| statistics | 0.000010 |
| preparing | 0.000007 |
| executing | 0.000177 |
| Sending data | 0.000012 |
| end | 0.000010 |
| query end | 0.000004 |
| closing tables | 0.000002 |
| removing tmp table | 0.000005 |
| closing tables | 0.000003 |
| freeing items | 0.000013 |
| cleaning up | 0.000021 |

```
17 rows in set, 1 warning (0.01 sec)
```

◦ 在学号上建立索引

- 先解释下索引。索引用来快速地寻找那些具有特定值的记录，所有MySQL索引都以B-树的形式保存。如果没有索引，执行查询时MySQL必须从第一个记录开始扫描整个表的所有记录，直至找到符合要求的记录。表里面的记录数量越多，这个操作的代价就越高。如果作为搜索条件的列上已经创建了索引，MySQL无需扫描任何记录即可迅速得到目标记录所在的位置。
- 聚集索引又叫主索引，其索引的排序方式和正文的排序方式一致。每个表只能有一个聚集索引，因为目录只能按照一种方法进行排序。用聚合索引比用不是聚合索引的主键速度快；用聚合索引比用一般的主键作order by时速度快，特别是在小数据量情况下。
- 相对应的，非聚集索引也叫辅助索引。
- 当数据量很小的时候，用聚集索引作为排序列要比使用非聚集索引速度快得明显的多；而数据量如果很大的话，如10万以上，则二者的速度差别不明显。此外，聚集索引插入数据时速度要慢（时间花费在“物理存储的排序”上，也就是首先要找到位置然后插入）。但是查询数据比非聚集数据的速度快。
- 由于学号是student的主键，所以根据以上分析，我们可以知道，当只有一个查询结果

时，这三种方法的快慢排序为：

- 非聚集索引 快于 聚集索引 快于 无索引。
- 对结果集中有多个元组的查询
 - 对于有多个元组的查询情况，差别不如3.1.1那么明显。三种情况差别都不大。
 - 聚集索引 > 非聚集索引 > 无索引。
- 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，在此以查询sc表中学号在'31404'和'31420'之间的学生的学号和课程号为例）分类似3.1.1的三种情况进行执行比较，注意系统处理的选择。
 - 对于查询条件为一个连续的范围的查询的情况，非聚集索引与无索引差别不是很大，不过聚集索引明显会快很多。
 - 总体上， 还是聚集索引 快于 非聚集索引 快于 无索引。
- 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。
 - 并非所有的情况索引都是会使速度变快的，比如我们往sc表里插入一条记录('001','C01','100')，同学在SQL Server下测试的执行时间结果为无索引用时2ms， 但是有索引38ms！！ 慢了非常多！

对相同查询功能不同查询语句的执行比较分析

- 比较有和没有group by的查询效率并分析

```
SELECT
    AVG(grade)
FROM
    SC
GROUP BY cno
HAVING cno = 'C01';
```

```
SELECT
    AVG(grade)
FROM
    SC
WHERE
```

```
cno = 'C01';
```

- 运行时间如下图,在有group by 的用时是0.0159.没有group by 的用时是0.0057,也就是说没有groupby,sql执行起来快一点

```
|          7 | 0.01590500 | select avg(grade) from sc group by cno having cno='C01'
|          |          |
|          8 | 0.00013500 | slect avg(grade) from sc where cno='C01'
|          |          |
|          9 | 0.00573500 | select avg(grade) from sc where cno='C01'
|          |          |
+-----+-----+-----+-----+-----+-----+-----+-----+
-----+
9 rows in set, 1 warning (0.00 sec)
```

- 在具体的数据对比中,最大的一项是optimizing

| | |
|----------------------|----------|
| starting | 0.000131 |
| checking permissions | 0.000009 |
| Opening tables | 0.000019 |
| init | 0.005897 |
| System lock | 0.000807 |
| optimizing | 0.000014 |
| statistics | 0.001727 |
| preparing | 0.000894 |
| Creating tmp table | 0.001968 |
| Sorting result | 0.000021 |
| executing | 0.000003 |
| Sending data | 0.002194 |
| Creating sort index | 0.002105 |
| end | 0.000022 |
| query end | 0.000020 |
| removing tmp table | 0.000007 |
| query end | 0.000004 |
| closing tables | 0.000006 |
| freeing items | 0.000035 |
| cleaning up | 0.000022 |

20 rows in set, 1 warning (0.00 sec)

[mysql> show profile for query 9;

| | |
|----------------------|----------|
| Status | Duration |
| starting | 0.000060 |
| checking permissions | 0.000005 |
| Opening tables | 0.000013 |
| init | 0.000022 |
| System lock | 0.000020 |
| optimizing | 0.005404 |
| statistics | 0.000028 |
| preparing | 0.000025 |
| executing | 0.000002 |
| Sending data | 0.000076 |
| end | 0.000003 |
| query end | 0.000005 |
| closing tables | 0.000007 |
| freeing items | 0.000022 |
| cleaning up | 0.000043 |

15 rows in set, 1 warning (0.01 sec)

- 比较以下两个查询，重写后的查询一定比原始查询更优吗，通过执行分析结果。

```
SELECT
    sno, sname, bdate
FROM
    student s1
```

```

WHERE
    bdate = (SELECT
              MAX(bdate)
            FROM
              student s2
            WHERE
              s1.dept = s2.dept);
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;

```

- 如下面的截图所示，我们可以看到没有重写时用时为0.02123850s。
 - 而重写用时为0.00326500s，如果加上创建table用时0.19003250和删除tmp用时0.08236775，则总共用时为0.27566525s。
 - 可见重写后，单纯的查询会比没重写的查询用时要少很多。甚至差了一个数量级。
 - 但是，如果没有对这个新的表进行多次相关的查询的话，那么建表时间会使得这个优化得不偿失。 <-差上面的截图--pagebreak->

查询优化

1. 查询选修了一门课的学生

- 由于学生不能选两次同样的课,因此我们可以根据总课程数做数值比较

```

mysql> explain select sno
[ -> from sc
[ -> group by sno
[ -> having count(cno)>=
[ -> (select count(cno) from course);

```

| id | select_type | table | partitions | type | possible_keys | key | key_len | ref | rows | filtered | Extra |
|----|-------------|--------|------------|-------|---------------|---------|---------|------|------|----------|-------------|
| 1 | PRIMARY | sc | NULL | index | PRIMARY | PRIMARY | 31 | NULL | 142 | 100.00 | Using index |
| 2 | SUBQUERY | course | NULL | index | NULL | PRIMARY | 11 | NULL | 7 | 100.00 | Using index |

```

2 rows in set (0.02 sec)

```

- 结果是:

```
[mysql> select sno from sc
[      -> group by sno
[      -> having count(cno)>=
[      -> (select count(cno) from course);
Empty set (0.01 sec)
```

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

- 思路:先选出一门,再从选中的里面选另一门的同学

```
[mysql> select sno from sc natural join course
[      -> where cname='数据库原理 ' and sno in
[      -> (select sno
[      -> from sc natural join course
[      -> where cname='操作系统 ');
```

```
+-----+
| sno   |
+-----+
| 31401 |
| 31403 |
| 31404 |
| 31406 |
| 31407 |
| 31408 |
| 31409 |
| 31411 |
| 31412 |
```

- 时间统计结果是


```
weixiao — mysql -u wx -p — 80x24
[mysql> show profile for query 1;
+-----+-----+
| Status                | Duration |
+-----+-----+
| starting               | 0.000099 |
| checking permissions   | 0.000035 |
| checking permissions   | 0.000003 |
| checking permissions   | 0.000001 |
| checking permissions   | 0.000004 |
| Opening tables         | 0.000018 |
| init                   | 0.000068 |
| System lock            | 0.000007 |
| optimizing             | 0.000026 |
| statistics             | 0.005687 |
| preparing              | 0.000060 |
| executing              | 0.000003 |
| Sending data           | 0.000224 |
| end                    | 0.000009 |
| query end              | 0.000010 |
| removing tmp table     | 0.000004 |
| query end              | 0.000002 |
| closing tables         | 0.000006 |
| freeing items          | 0.000028 |
| cleaning up            | 0.000014 |
```

实验反思

`show profiles;`

```
[mysql> show profiles;
Empty set, 1 warning (0.00 sec)
```

- 解决: `show profiles`为空则说明数据库关闭了profile功能,使用下面的代码即可成功开启

```
mysql> SET profiling=1;
```

心得体会

- 本次实验主要完成了查询分析和查询优化.mysql中对于同一个查询结果的不同语法,他们的速度存在天差地别,这次实验让我们知道在SQL中有各种各样的手段去查询时间,去完成优化,并且利用profile信息或者explain语句来做查询效率分析,其中各个统计值的含义在MySQL手册上均可以查到,其中也不乏一点晦涩难懂的参数.
- 了解并学习各种SQL优化方法和原理,可以在处理大型查询和复杂查询时,加快查询的效率,更快更高效的查出结果.此外,对索引的理解和使用也可以用来提高数据库的查询效率,他们的主要思想就是类似于书本目录的方式,使得一个查询过程未必需要遍历全表,而是可以利用一些搜索码进

行快速定位和查找.