2015-2016 中山大学本科生实验报告

科目：数据库系统实验　　**教师**：郑贵锋

年级	13 级	专业（方向）	移动信息工程
学号	13354485	姓名	朱琳
电话	13726231932	Email	280273861@qq.com

一 实验目的

1.了解索引，理解索引在查询之中起到的优化作用。

二 实验题目以及实验结果

1.首先导入数据，创建数据库名为'SYSU'，将 lab10.ddl 与 lab10.tbl 放到某一确定目录中（建议目录名中无中文），使用一下指令分别导入表与表中数据。表格结构为：

名字	身份	节目	周年	到场时间	地点

```
mysql> source c:/users/sauce/desktop/db/lab10/lab10.ddl
Query OK, 0 rows affected (0.08 sec)
```

```
mysql> load data local infile 'c:/users/sauce/desktop/db/lab10/lab10.tbl' into t
able anniversary fields terminated by '|' lines terminated by '|\r\n';
Query OK, 100000 rows affected (2.71 sec)
Records: 100000  Deleted: 0  Skipped: 0  Warnings: 0
```

【结果】

```
mysql> create database lab10;
Query OK, 1 row affected (0.24 sec)

mysql> use lab10;
Database changed
mysql> source e:/Desktop/lab10/lab10.ddl
Query OK, 0 rows affected (1.09 sec)

mysql> load data local infile 'e:/Desktop/lab10/lab10.tbl' into table anniversar
y fields terminated by '|' lines terminated by '|\r\n';
Query OK, 100000 rows affected (5.37 sec)
Records: 100000  Deleted: 0  Skipped: 0  Warnings: 0
```

【查看下表的结构】

```
mysql> describe  anniversary;
+-------------+----------+------+-----+---------+-------+
| Field       | Type     | Null | Key | Default | Extra |
+-------------+----------+------+-----+---------+-------+
| name        | char(20) | YES  |     | NULL    |       |
| identity    | char(15) | YES  |     | NULL    |       |
| performance | char(20) | YES  |     | NULL    |       |
| num         | int(11)  | YES  |     | NULL    |       |
| time        | time     | YES  |     | NULL    |       |
| place       | char(7)  | YES  |     | NULL    |       |
+-------------+----------+------+-----+---------+-------+
6 rows in set (0.18 sec)
```

2.查询一下内容，记录查询时间。

(1)参加了 **90** 周年庆的所有人员以及他们的身份。

Mysql 语句：select name,identity from anniversary where num=90;

【**结果**】因为数据太多，只截取最后的结果，看出是 1104 行，0.12s。

```
1104 rows in set (0.36 sec)
```

(2)在东校区 **50** 周年庆上参加唱歌活动的所有人员。

Mysql 语句：select name,identity from anniversary where num=50 and place="East" and performance="Singing";

【结果】因为数据太多，只截取最后的结果，看出是 21 行，0.11s。

```
21 rows in set (0.11 sec)
```

(3)连续参加了珠海校区 **89** 和 **90** 周年庆的所有人员以及身份。

```
mysql> select name,identity from anniversary where num=89 and place="zhuhai" and
 name in (select name from anniversary where  num=90 and place="zhuhai");
+-------------+--------------+
| name        | identity     |
+-------------+--------------+
| Person35091 | HonoredGuest |
+-------------+--------------+
1 row in set (0.40 sec)
```

(4)参加了南校区 **85** 周年庆并进行了两项以上节目表演的所有人员。

```
mysql> select name from anniversary where num=85 and place="South" group by name
 having count(*)>1;
Empty set (0.26 sec)
```

3. 在 **num** 上建立索引，重做步骤 **2**，对比时间区别。

【建立索引】

```
mysql> CREATE INDEX numIndex ON anniversary (num) ;
Query OK, 0 rows affected (1.33 sec)
Records: 0  Duplicates: 0  Warnings: 0
```

(1)参加了 **90** 周年庆的所有人员以及他们的身份。

Mysql 语句：select name,identity from anniversary where num=90;

【**结果**】因为数据太多，只截取最后的结果，看出是 1104 行，0.12s。

```
1104 rows in set (0.19 sec)
```

(2)在东校区 50 周年庆上参加唱歌活动的所有人员。

Mysql 语句：select name,identity from anniversary where num=50 and place="East" and performance="Singing";

【结果】因为数据太多，只截取最后的结果，看出是 21 行，0.11s。

```
21 rows in set (0.11 sec)
```

(3)连续参加了珠海校区 89 和 90 周年庆的所有人员以及身份。

```
mysql> select name,identity from anniversary where num=89 and place="zhuhai" and
 name in (select name from anniversary where  num=90 and place="zhuhai");
+-------------+--------------+
| name        | identity     |
+-------------+--------------+
| Person35091 | HonoredGuest |
+-------------+--------------+
1 row in set (0.28 sec)
```

(4)参加了南校区 85 周年庆并进行了两项以上节目表演的所有人员。

```
mysql> select name from anniversary where num=85 and place="South" group by name
 having count(name)>1;
Empty set (0.16 sec)
```

4. 插入、删除、更新，记录下时间（步骤 3 中建立的索引不删除）。

(1)将名字为 Person1 的 num 修改为 80。

```
mysql> update anniversary set num=80 where name="Person1";
Query OK, 3 rows affected (0.47 sec)
Rows matched: 3  Changed: 3  Warnings: 0
```

(2)将名字为 Person1 的人删除。

```
mysql> delete from anniversary where name="Person1";
Query OK, 3 rows affected (0.59 sec)
```

(3)插入一条记录(Peter, Actor, Singing, 90, 15:40, Zhuhai).

```
mysql> insert into anniversary values("Peter","Actor","Singing",90,"15:40","Zhuh
ai");
Query OK, 1 row affected (0.08 sec)
```

5. 把表 drop 掉，重新载入，重做步骤 4，对比时间。

```
mysql> drop table anniversary;
Query OK, 0 rows affected (0.28 sec)

mysql> show tables;
Empty set (0.00 sec)
```

```
mysql> load data local infile 'e:/Desktop/lab10/lab10.tbl' into table anniversar
y fields terminated by '|' lines terminated by '|\r\n';
Query OK, 100000 rows affected (3.38 sec)
Records: 100000  Deleted: 0  Skipped: 0  Warnings: 0
```

(1)将名字为 Person1 的 num 修改为 80。

```
mysql> update anniversary set num=80 where name="Person1";
Query OK, 3 rows affected (0.43 sec)
Rows matched: 3  Changed: 3  Warnings: 0
```

(2)将名字为 Person1 的人删除。

```
mysql> delete from anniversary where name="Person1";
Query OK, 3 rows affected (0.57 sec)
```

(3)插入一条记录(Peter, Actor, Singing, 90, 15:40, Zhuhai).

```
mysql> insert into anniversary values("Peter","Actor","Singing",90,"15:40","Zhuh
ai");
Query OK, 1 row affected (0.10 sec)
```

6. 设计一个组合索引，要求比步骤 2 中每一项要更快。

```
mysql> create index twoindex on anniversary(num,place);
```

(1)参加了 90 周年庆的所有人员以及他们的身份。

```
1105 rows in set (0.25 sec)
```

(2)在东校区 50 周年庆上参加唱歌活动的所有人员。

```
21 rows in set (0.05 sec)
```

(3)连续参加了珠海校区 89 和 90 周年庆的所有人员以及身份。

```
mysql> select name,identity from anniversary where num=89 and place="zhuhai" and
 name in (select name from anniversary where  num=90 and place="zhuhai");
+-------------+--------------+
| name        | identity     |
+-------------+--------------+
| Person35091 | HonoredGuest |
+-------------+--------------+
1 row in set (0.14 sec)
```

(4)参加了南校区 85 周年庆并进行了两项以上节目表演的所有人员。

```
mysql> select name from anniversary where num=85 and place="South" group by name
 having count(*)>1;
Empty set (0.07 sec)
```

三 实验结果对比

1.第二步和第三步以及第六步时间上的区别

	第一小题	第二小题	第三小题	第四小题
不加索引	0.36	0.11	0.40	0.26
加上 num 索引	0.19	0.11	0.28	0.22
组合索引 num,place	0.25	0.05	0.14	0.07

【分析】做了很多遍，才做到理想的效果，虽然也不是完美。这都是建立索引之后第一次运行的结果。若建立索引之后，运行重复的语句，第一次运行以后会非常快。具体的在【实验感想】部分详细描述。

2.第四步和第五步时间上区分

	更新	删除	插入
有 num 索引	0.47	0.59	0.08
不加索引	0.43	0.57	0.10

【分析】从上述时间对比上可以看出加了索引好像并没有什么用，因为 where 里面根本没有用到 nun 这个属性，所以也用不到索引。之所以有索引的时候慢一点可能是因为 B+树是个动态的索引，我们增删改数据之后索引也要进行相应的调整，但是对于这种只有一个语句的这些操作，影响是微乎其微的。

三 实验感想

1.这次实验看起来很简单的样子，但是在做的过程中还是有些许的混乱，本来做了一遍感觉没什么问题了，结果还是出现了一些毛病什么的。

2.关于第四小问那个表演节目两项以上的人什么的，我最初的思路是知道使用 group by，然后再 COUNT，然后翻了下之前做的关于 Group by 这个函数的那次实验，发现跟这次的有很大的差别，然后我就有些蒙圈。后来才知道原来需要使用 having，真是才疏学浅，之前竟然不知道还有这么个语句。通过查阅资料得知，having 字句可以让我们筛选成组后的各种数据，where 字句在聚合前先筛选记录。而 having 子句在聚合后对组记录进行筛选。总之就是 having 一般跟在 group by 之后了，其他的情况用 where 基本都可以解决。

3.最坑的就是，添加索引之后，如果第一次执行一个语句，那么运行时间还是正常的，但是第二次调用就变得超级快超级快，具体看如下截图：
以下是添加索引之后第二次或者以上执行语句：

```
1104 rows in set (0.02 sec)
```

```
21 rows in set (0.02 sec)
```

```
mysql> select name,identity from anniversary where place="zhuhai" and num=89 an
 name in (select name from anniversary where place="zhuhai" and num=90);
+-------------+--------------+
| name        | identity     |
+-------------+--------------+
| Person35091 | HonoredGuest |
+-------------+--------------+
1 row in set (0.09 sec)
```

```
mysql> select name from anniversary where num = 85 and place = "South" group by
name having count(*) > 1;
Empty set (0.02 sec)
```

原来都是这样：

加上 num 索引	0.19	0.11	0.28	0.22

前后差别非常大。经过反复验证，都是在添加索引之后的第二次以后运行速度变快的。但是奇怪的是，如果不添加索引，这种现象并不存在。就是说，在没有添加索引的情况下，重复运行指令并不会提升运行速度。

考虑到 mysql 可能内置了 cache，但是没添加索引的时候竟然没有 cache 现象，这很奇怪啊。并不知道 mysql 对他们做了什么。。而且运行时间也总是抽风。心真是累。