第一次数据库实验

# 1 测试环境

|  |  |
| --- | --- |
| 软件环境 |  |
| 操作系统 | **Windows 7 32bit** |
| MySQL版本 | **5.5.19** |
| 存储引擎 | **Memory/MyISAM** |

|  |  |
| --- | --- |
| 硬件环境 |  |
| CPU | **Intel Core i3 3.10GHz 3.10GHz** |
| 内存 | **4GB** |

# 2 建表

创建用于测试的表结构，默认使用MyISAM引擎，同时根据速度需求，使用Memory引擎进行比较测试。在my.ini中改变max\_heap\_table\_size大小（默认是16M，改为256M），建表语句如下：

|  |
| --- |
| delimiter $$  CREATE TABLE `test` (  `ID` int(10) unsigned NOT NULL AUTO\_INCREMENT,  `col1` varchar(5) NOT NULL,  `col2` char(5) NOT NULL,  `col3` tinyint(4) NOT NULL,  PRIMARY KEY (`ID`),  KEY `idx\_col2` (`col2`)  ) ENGINE=Memory AUTO\_INCREMENT=1 DEFAULT CHARSET=latin1$$ |

# 3 插入数据

## 3.1 使用导入数据的方法

使用导入数据的方法是最快的方法。

1. 使用java语言生成1000000条数据输出到txt.data文件.
2. 使用“LOAD DATA LOCAL INFILE ‘D:\\test.data‘ INTO TABLE test;”将数据导入到数据库中

## 3.2 使用函数插入

根据实验要求，插入100万条数据，语句如下：

|  |
| --- |
| delimiter $$  CREATE DEFINER=`root`@`localhost` PROCEDURE `test\_insert`(iNum MEDIUMINT )  BEGIN  DECLARE iFlag TINYINT DEFAULT 0;  WHILE iNum>0  DO  IF iFlag=0 THEN  START TRANSACTION;  END IF;  INSERT INTO test(col1,col2,col3)  VALUES(**getname(),ran\_string()**,floor(1+rand()\*20));  SET iFlag=iFlag+1;  IF iFlag=100 THEN  COMMIT;  SET iFlag=0;  END IF;  SET iNum=iNum-1;  END WHILE;  END$$ |

其中getname()如下所示，作用是产生第二列的随机姓名

|  |
| --- |
| delimiter $$  CREATE DEFINER=`root`@`localhost` FUNCTION `getname`() RETURNS char(5) CHARSET latin1  BEGIN  declare counter int unsigned default 0;  declare result char(5) default '';  set counter=floor(rand()\*10);  if counter = 0 then  set result = 'Mike';  elseif counter = 1 then  set result = 'Bob';  elseif counter = 2 then  set result = 'Jack';  elseif counter = 3 then  set result = 'Alice';  elseif counter = 4 then  set result = 'Cathy';  elseif counter = 5 then  set result = 'Ann';  elseif counter = 6 then  set result = 'Betty';  elseif counter = 7 then  set result = 'Cindy';  elseif counter = 8 then  set result = 'Mary';  else  set result = 'Jane';  end if;  return result;  END$$ |

randstring()方法如下表所示，随机产生一个5位字母，字母限制在a-e

|  |
| --- |
| delimiter $$  CREATE DEFINER=`root`@`localhost` FUNCTION `ran\_string`() RETURNS char(5) CHARSET latin1  BEGIN  declare counter int unsigned default 0;  declare const\_char char(5) default 'abcde';  declare result char(5) default '';  while counter<5 do  set result=concat(result,substr(const\_char,ceil(rand()\*4),1));  set counter=counter+1;  end while;  return result;  END$$ |

# 4 优化查询速度

1. 锁定test表为写LOCK TABLES test WRITE; 解锁 UNLOCK TABLES;
2. 关闭自动提交 SET AUTOCOMMIT = 0;
3. 关闭索引ALTER TABLE test DISABLE KEYS;
4. 更改key\_buffer\_size***，***用于索引块的缓冲区大小，增加它可得到更好处理的索引，但是会影响系统转换页的速度；

# 5 查询数据

## 步骤三的查询语句。

|  |
| --- |
| select count(\*) from test group by col1 order by count(\*); |

## 步骤四的查询语句，将结果输出到output.txt文件中。

|  |
| --- |
| select \* INTO OUTFILE ‘C:\\output.txt’ from test where col2 like 'ab%' |

## 步骤五的测试

### 使用mysql自带的并发测试工具，针对myisam和memory引擎进行并发测试。

|  |
| --- |
| mysqlslap -a --concurrency=5 --iterations=1 --number-of-queries 1000--engine=myisam,memory --debug-info -uroot -p |

### 使用Java语言编写多线程操作，模拟并发执行

编写查询语句

|  |
| --- |
| **public** **static** **void** query(String str) {  **try** {  System.*out*.println("start "+ str+ ": 'select count(\*) from test group by col1 order by count(\*);'");  Long beginTime = System.*currentTimeMillis*();  Statement st = *conn*.createStatement();  String sql = "select count(\*) from test group by col1 order by count(\*);";  st.executeQuery(sql);  System.*out*.println(str + " time: " + (**double**) (endTime - beginTime)/ 1000 + " s");  st.close();  } **catch** (Exception e) {  e.printStackTrace();  }  } |

使用Java多线程模拟并发.

|  |
| --- |
| //使用线程池  ExecutorService service = Executors.*newCachedThreadPool*();  // 模拟5个客户端访问  **for** (**int** index = 1; index < =*client\_num*; index++) {  **final** **int** NO = index;  Runnable runnable = **new** Runnable() {  @Override  **public** **void** run() {  // **TODO** Auto-generated method stub  **try** {  System.*out*.println("Client: " + NO);  *query*("Client: " + NO);  } **catch** (Exception e) {  // **TODO**: handle exception  e.printStackTrace();  }  }  };  service.execute(runnable);  }  service.shutdown(); |

# 6 测试结果

## 6.1 步骤二，插入执行时间

### 使用导入数据的方法

使用MyISAM引擎



使用Memory引擎



### 使用插入的方法

使用MyISAM引擎

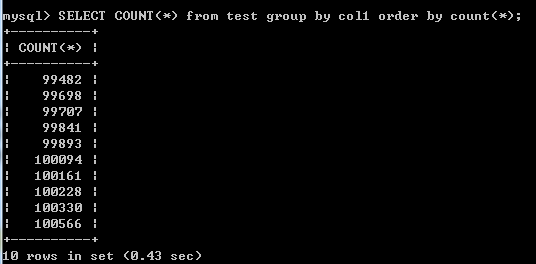


使用Memory引擎

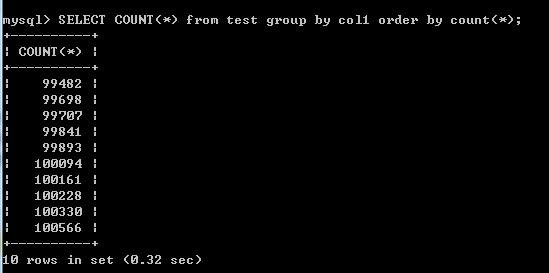


## 6.2 步骤三，查询执行时间

使用MyISAM引擎



使用Memory引擎



## 6.3 步骤四，查询执行时间

使用MyISAM引擎



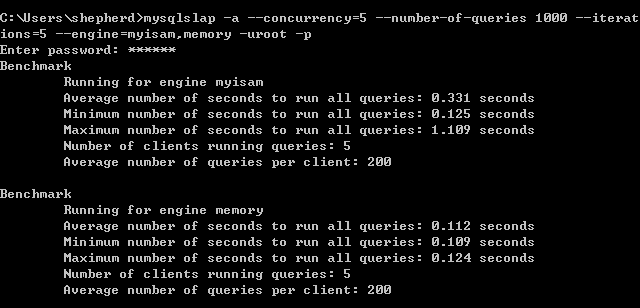
使用Memory引擎



输出结果见output.txt

## 6.4 步骤五，并发执行效果

### 使用mysql自带的并发测试工具mysqlslap测试并发性，比较结果如下：



### 使用Java语言编写多线程操作，模拟并发执行

|  |
| --- |
| Client: 1  start Client1: 'select count(\*) from test group by col1 order by count(\*);'  Client: 2  start Client2: 'select count(\*) from test group by col1 order by count(\*);'  Client: 4  start Client4: 'select count(\*) from test group by col1 order by count(\*);'  Client: 5  start Client5: 'select count(\*) from test group by col1 order by count(\*);'  Client: 3  start Client3: 'select count(\*) from test group by col1 order by count(\*);'  Client1 time: 0.572 s  Client3 time: 1.115 s  Client5 time: 1.668 s  Client4 time: 2.226 s  Client2 time: 2.773 s |

# 7 讨论

为了追求速度，使用了Memory引擎的内存表，一般的说来，使用内存表可以比MyISAM表快一个数量级。但是由于内存表的数据都保存在内存中，所以相关查询根本不需要为此等待磁盘I/O处理。但是一旦系统重启，内存表的表结构仍然会保存，但是相关数据都将会丢失。

于此同时，内存表使用表级加锁，只支持较低的写并发，也不支持TEXT或BLOB字段类型。它们只支持固定大小行，比如将VARCHAR实际存储为CHAR类型，可能会浪费不少内存。

为了处理这些问题，可以使用复制搭建，创建一个复制机器来确保数据安全，也可以使用数据备份，也可以通过改变表结构，如将Memory引擎的表改变为MyISAM引擎的表。