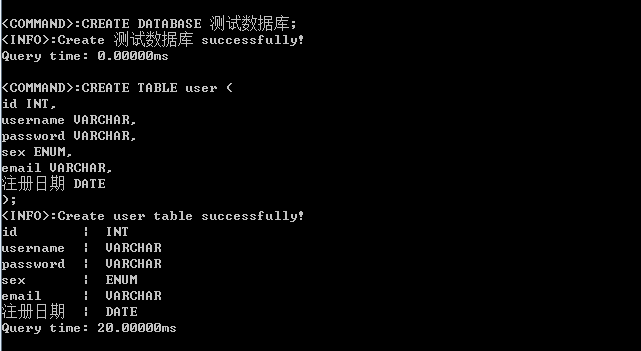
Mini-db测试报告

IT不离线

2016/9/12

**功能测试**

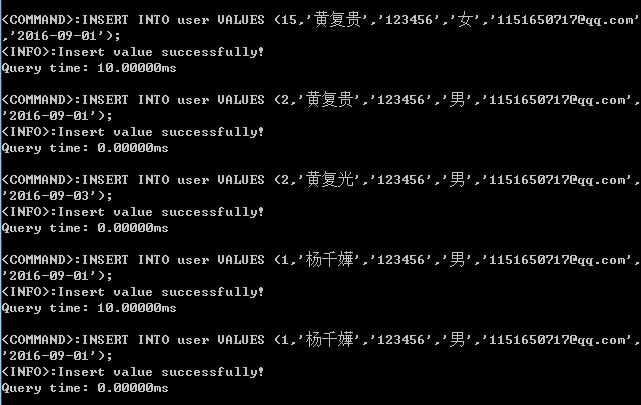
1. 建库建表。

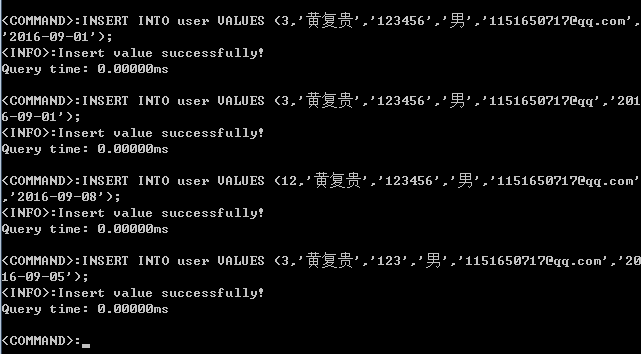


在桌面创建了一个数据库文件夹，里面有一张数据库表，记录了表头信息。



1. 插入数据。





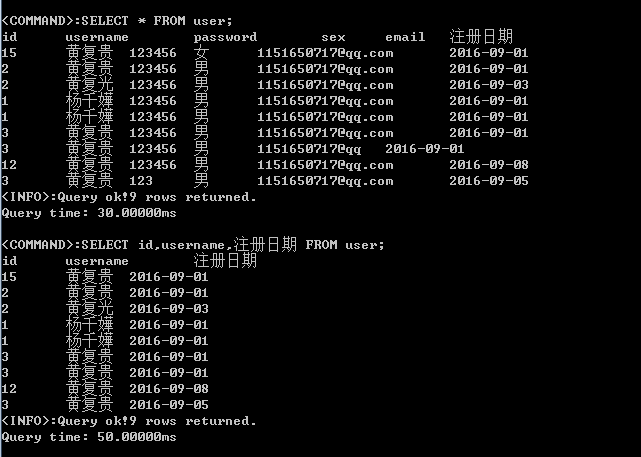
运行结果：

磁盘文件中实时持久化了数据。

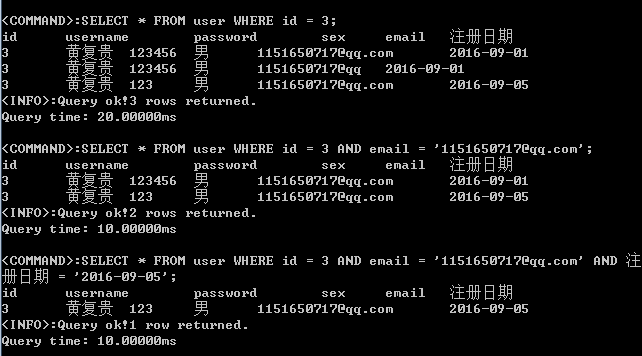


1. 查找数据。

查找表中全部数据，指定字段查找：



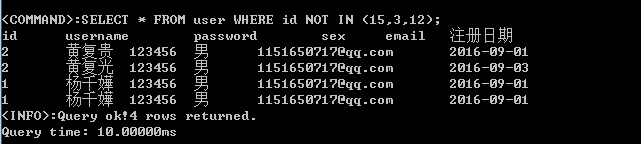
指定WHERE多条件与查找：



指定WHERE多条件或查找：



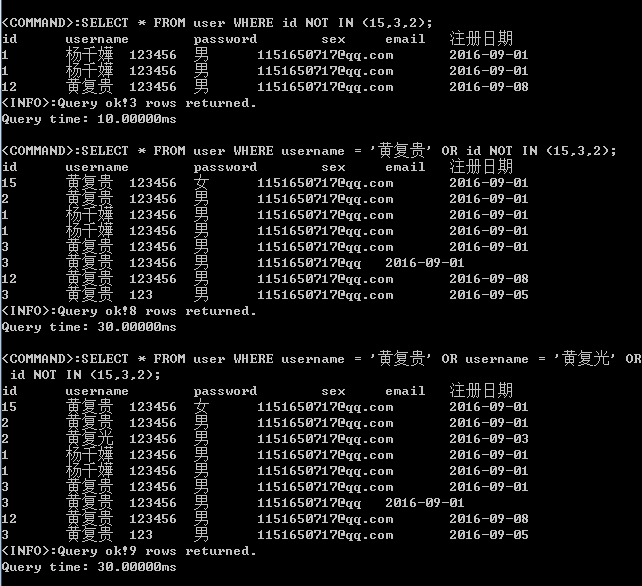
指定WHERE非条件查找：

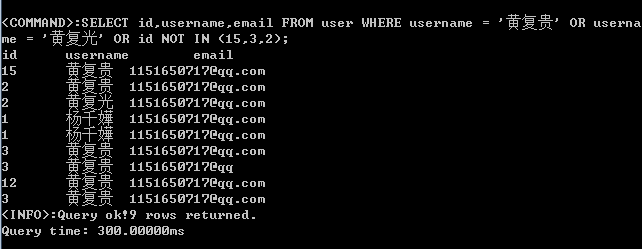


与非连用：

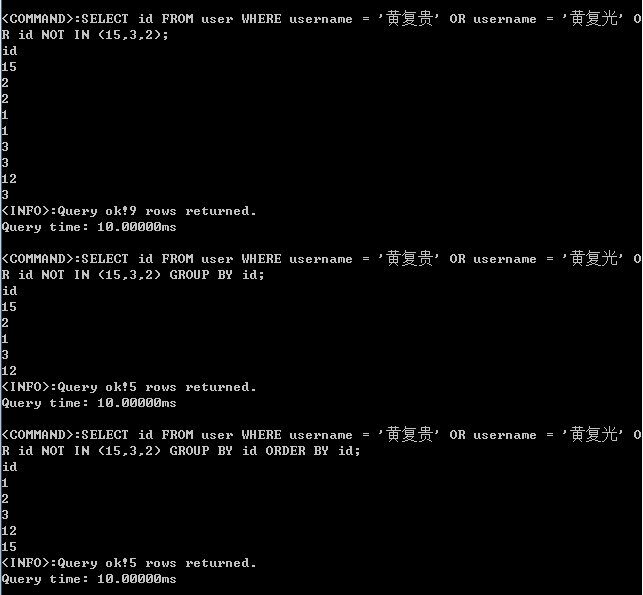


或非连用：

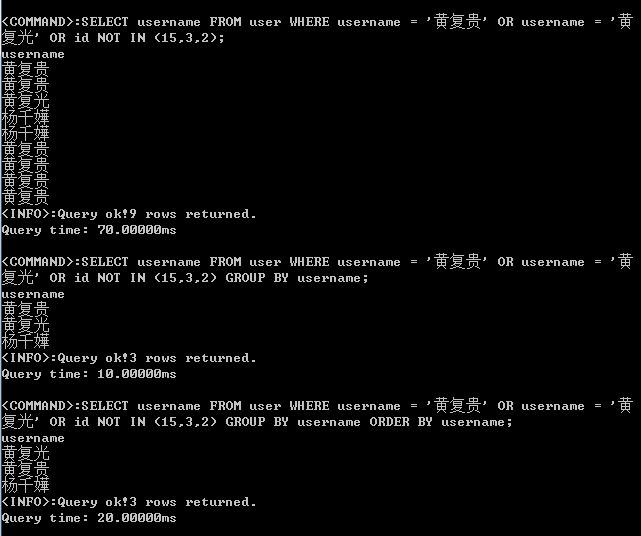




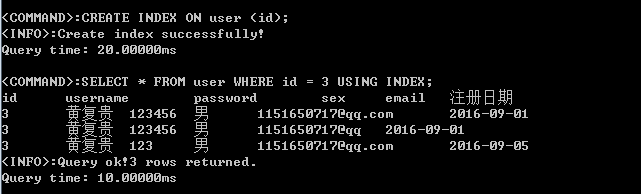
数值类型的GROUP BY单用，与ORDER BY连用：



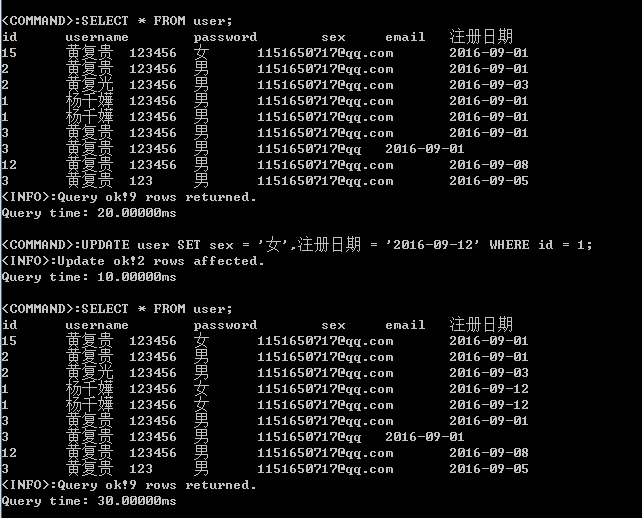
字符型的GROUP BY单用，与ORDER BY连用：



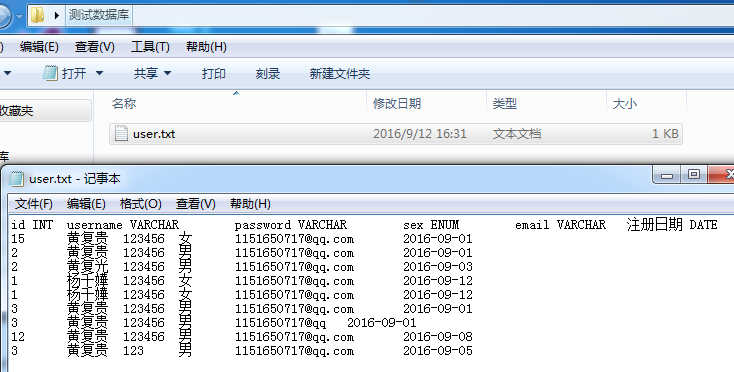
1. 创建索引与使用索引进行查找。



1. 更新数据。



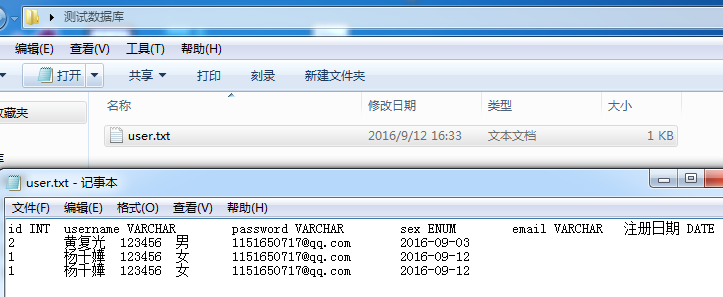
数据实时同步至磁盘：



1. 删除数据。



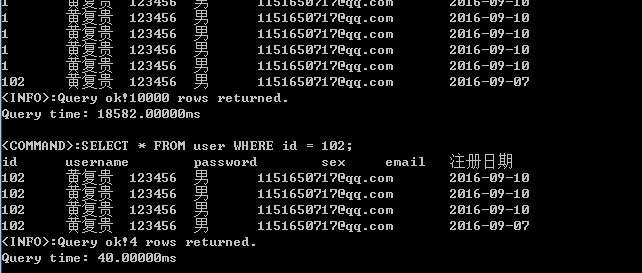
数据实时同步至磁盘：



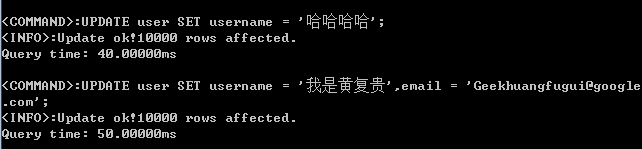
**性能测试**

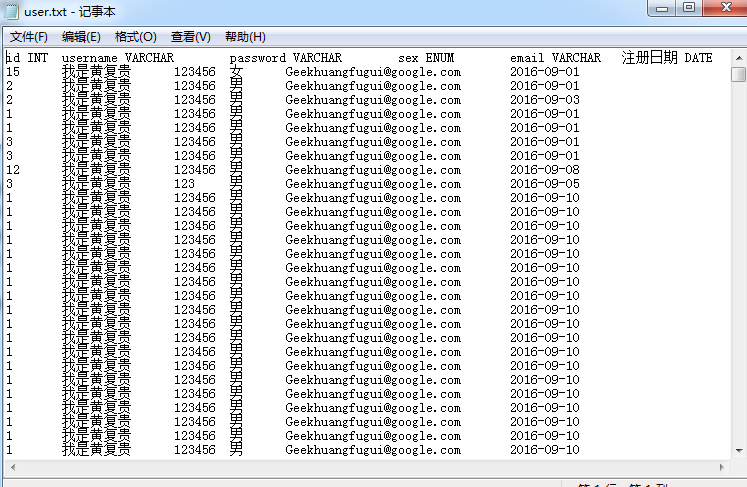
事先说明：项目数据库不是内存数据库，即需要用到数据时才从磁盘文件中读取，也不会将表数据存储在内存中。因此只要没有用到索引数据结构，每个操作都是全盘扫描（while(fgets(buffer,1024,file)!=NULL)）。

user数据表中有10000条数据，其中id为102的记录有4条，现在要查出这4条记录，如下：



从结果可以看到，在磁盘10000条记录中查找4条记录（FTS，从磁盘中一行行读入，再进行相应的逻辑判断），时间为40ms。相信UPDATE也是几乎一样的时间。

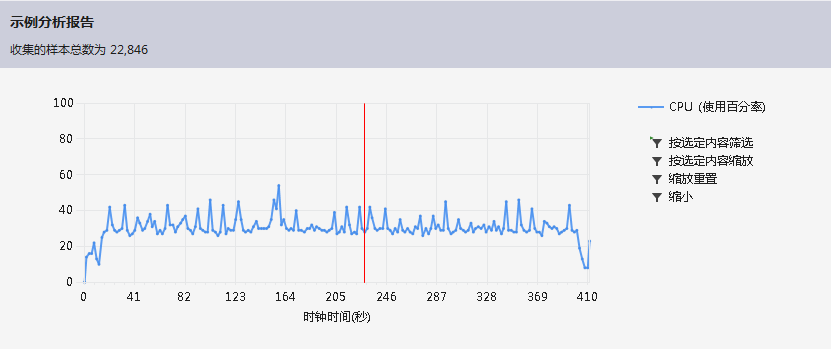


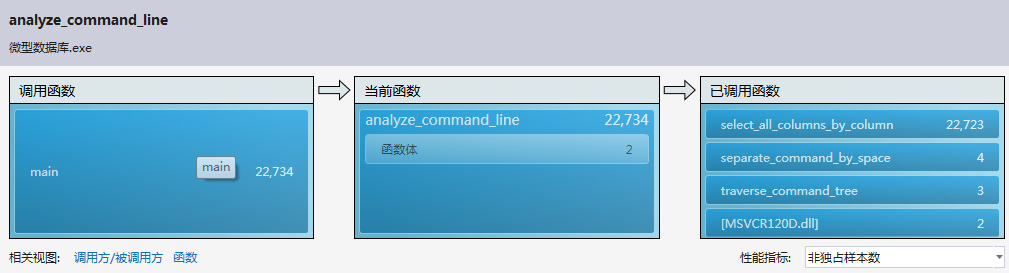


总结（一）：经过多次多种SELECT语句的测试，对于10000条数据的全盘扫描再进行各种各样的逻辑判断，时间几乎都在20ms~60ms范围内。

压力下性能诊断测试：

制造压力将10000条SELECT \* FROM user WHERE id = 102;语句同时复制到.exe执行（O(10000\*10000)），查看这段时间内程序内部性能。



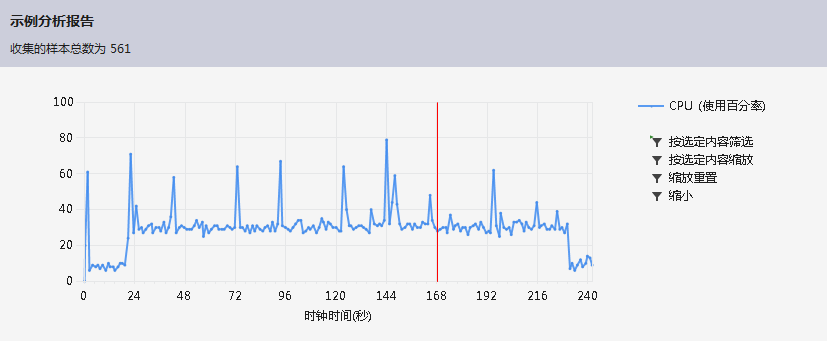


可以看到：10000次执行SELECT \* FROM user WHERE id = 102; （O(10000\*10000)）,总共花费22734ms，即差不多23秒的时间。

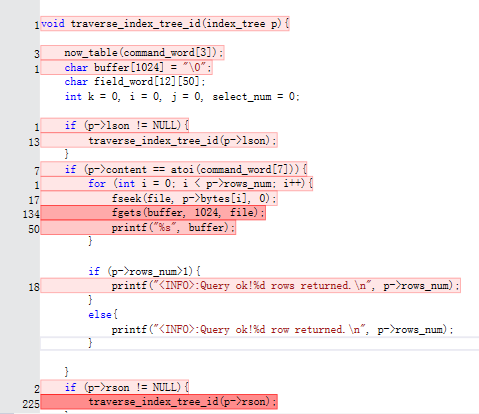


其中，净是从磁盘文件中读取数据就花费了17162ms，因此程序大部分的执行时间不是逻辑判断，而是从磁盘中读取数据至内存（10000\*10000次fgets）。

为id字段建立索引数据结构后，优化查找，执行一万次SELECT \* FROM user WHERE id = 102 USING INDEX;同样是查找相同的结果。分析诊断性能：







可以看到：建立索引后，不再进行全盘扫描将磁盘文件内容一行行读入内存再进行逻辑判断。直接遍历索引获得满足条件的记录在磁盘中的位置再直接读取。10000次SELECT \* FROM user WHERE id = 102 USING INDEX;的执行时间为447ms。从原本的22734ms到现在的447ms。时间效率提高了50倍。

总结（二）：对于“外存”数据库而言，建立索引不再进行全盘扫面后，直接定位记录在表中的位置，能很大程度优化查找速度。