|  |
| --- |
| 哈尔滨工业大学(深圳) |
| **《数据库》实验报告** |
|  |
| **实验五**  **查询处理算法的模拟实现**  学 院: 计算机科学与技术   |  |  | | --- | --- | | 姓 名: | 王铭 | | 学 号: | 190110509 | | 专 业: | 计算机科学与技术 | | 日 期: | 2021-11-2 | |

# 实验目的

①理解索引的作用

②掌握关系选择、投影、连接、集合的交、并、差等操作的实现算法

③加深对算法I/O复杂性的理解

# 实验环境

编程语言：C

编译环境：Code::Blocks 17.12

# 实验内容

基于由C语言编写的ExtMem程序库，模拟读写磁盘的过程，利用有限的内存，实现关系选择、连接操作算法。实现集合并、交、差操作算法。

具体内容：

①实现基于线性搜索的关系选择算法：基于ExtMem程序库，使用C语言实现线性搜索算法，选出S.C=130的元组，记录IO读写次数，并将选择结果存放在磁盘上。

② 实现两阶段多路归并排序算法（TPMMS）：利用内存缓冲区将关系R和S分别排序，并将排序后的结果存放在磁盘上。

③ 实现基于索引的关系选择算法：利用（2）中的排序结果为关系R或S分别建立索引文件，利用索引文件选出S.C=130的元组，并将选择结果存放在磁盘上。记录IO读写次数，与（1）中的结果对比。

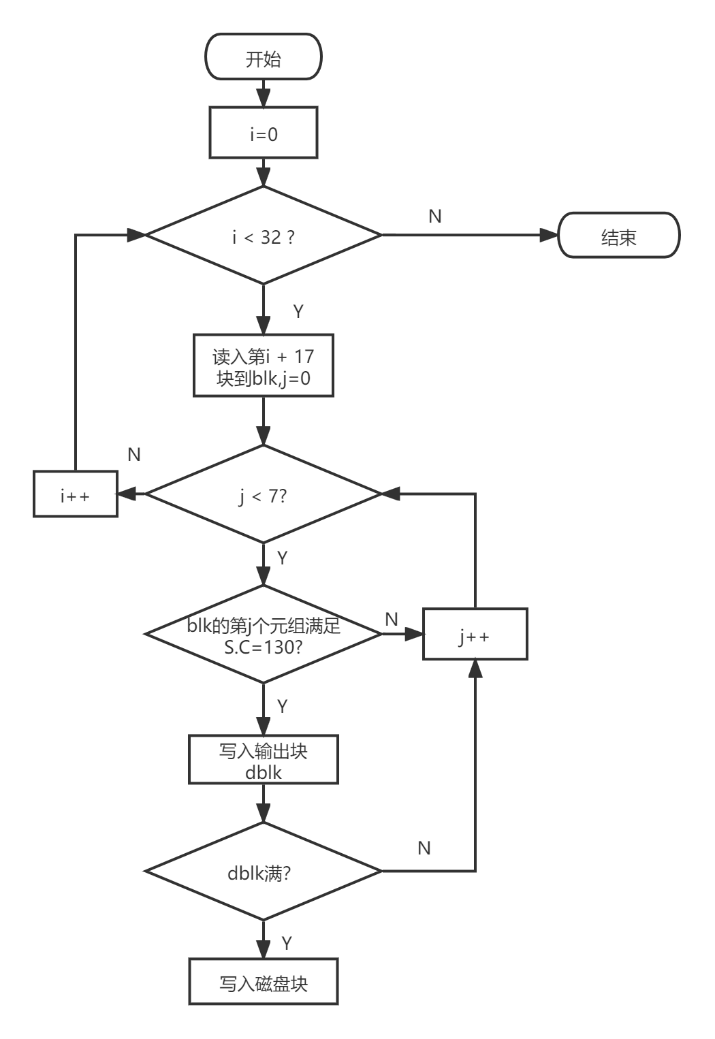
④ 实现基于排序的连接操作算法（Sort-Merge-Join）：对关系S和R计算S.C连接R.A ，并统计连接次数，将连接结果存放在磁盘上。

⑤ 实现基于排序或散列的两趟扫描算法，实现其中一种集合操作算法：并（S R）、交（S R）、差（S - R）中的一种。将结果存放在磁盘上，并统计并、交、差操作后的元组个数。

# 实验过程

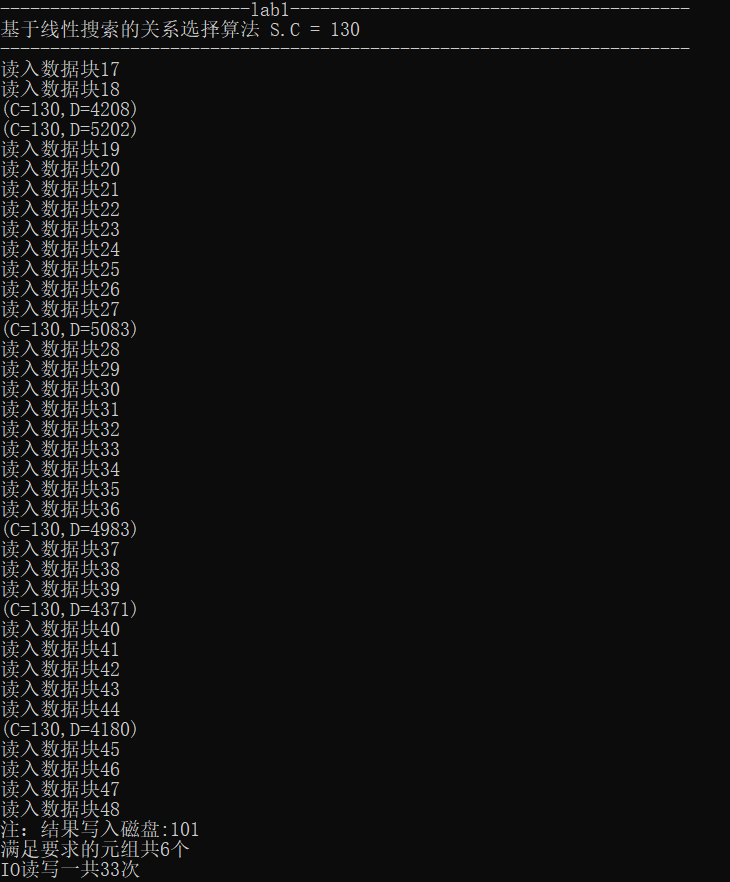
1. **实现基于线性搜索的关系选择算法**

问题分析：采用线性搜索的方法，依次读入存储关系S的磁盘块17到48到内存的输入块中，每个磁盘块中包含7条记录，则对每一条记录的第一个属性（S.C）判断，若满足S.C=130，则将该条记录写入到输出块中。输出块装满七条记录时，则写入到磁盘中保存即可。查询结束后，若输出块中仍有数据但未满七条，也要写入磁盘块。流程图如下：



题目1流程图

实验结果：



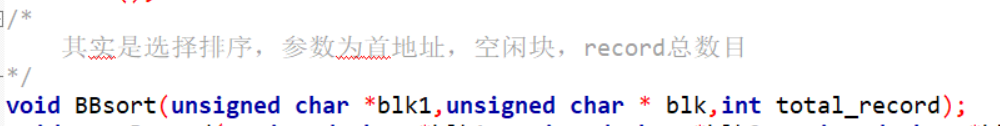
题目1结果截图

1. **实现两阶段多路归并排序算法（TPMMS）**

问题分析：扫描磁盘块两遍对磁盘块中的数据进行排序，第一趟扫描进行组的划分，实现每个组的内排序，第二趟扫描进行归并排序，将每一个有序的组归并成一个整体有序集合并写回到磁盘中去。

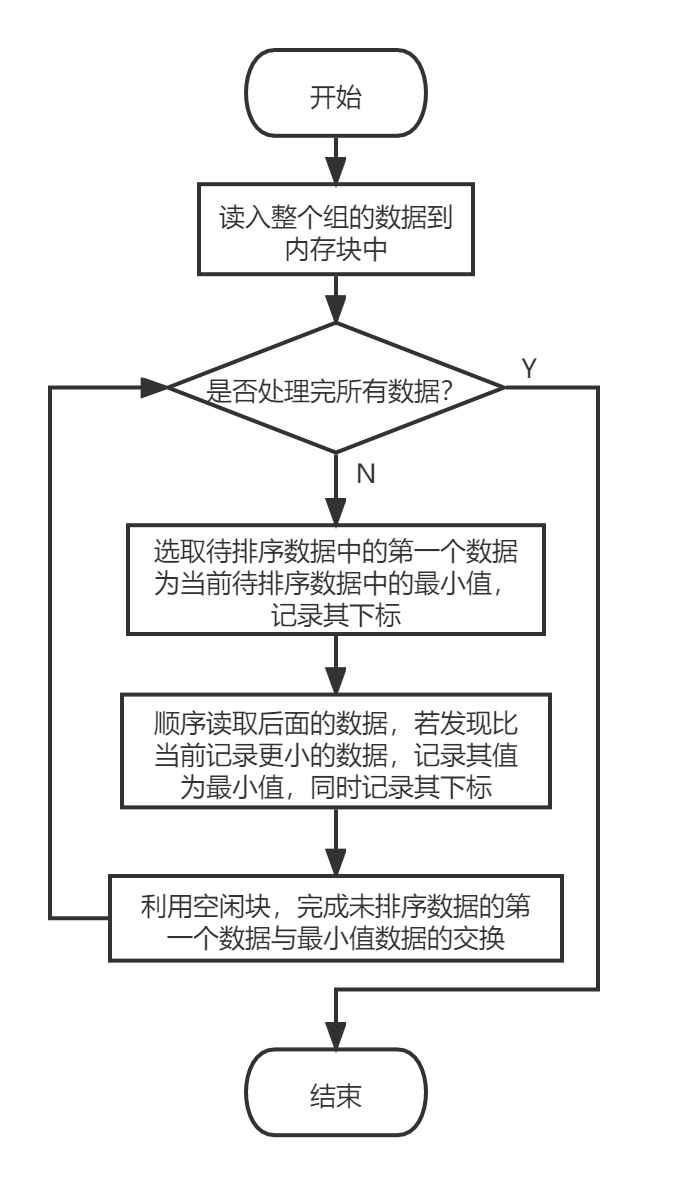
第一趟排序，即将分好的组整个读到内存中，利用选择排序进行组内排序，并保存到磁盘中。其中，R每组3个块，最后一个组1一个块，共六组。S每组6个块，最后一组2个块，共六组。将该内排序过程封装成函数BBsort()，可以大幅度减少代码量。

BBsort的函数声明如下:



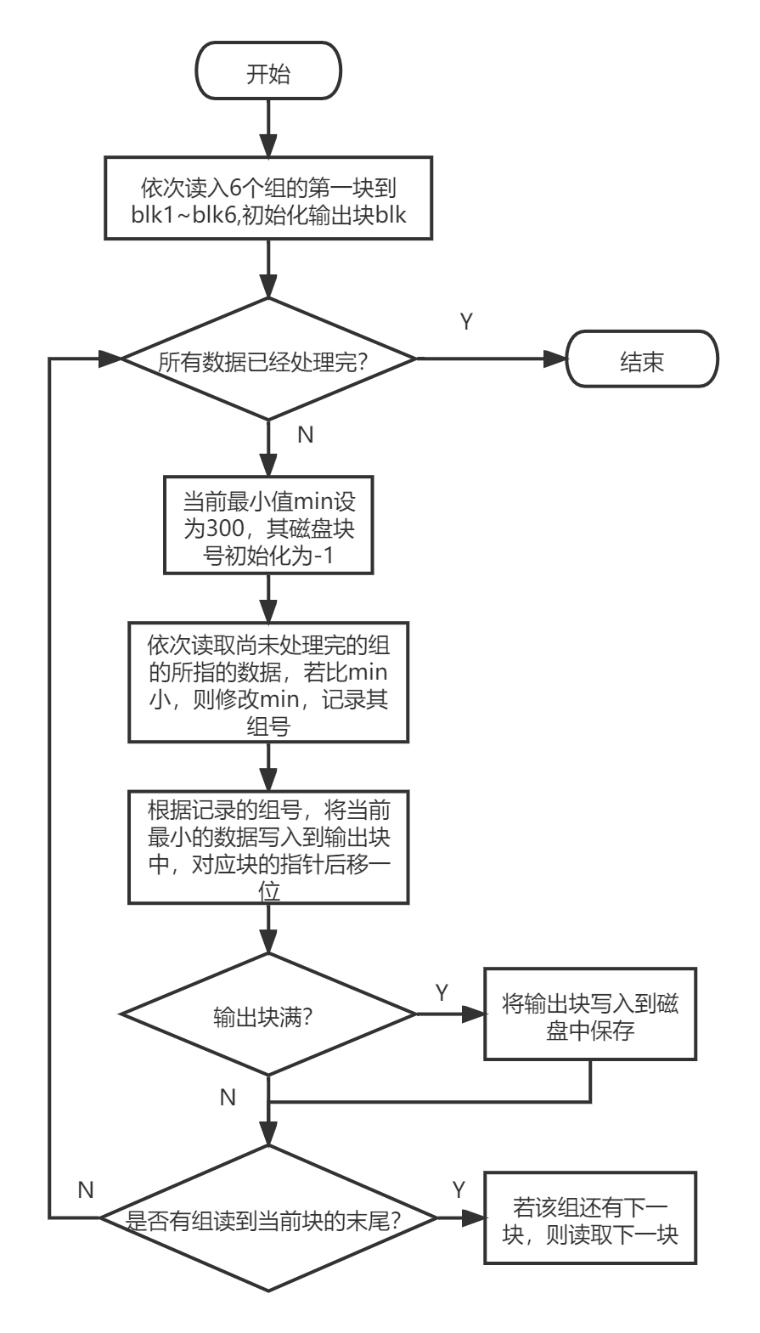
BBsort函数声明

该算法流程图如下：



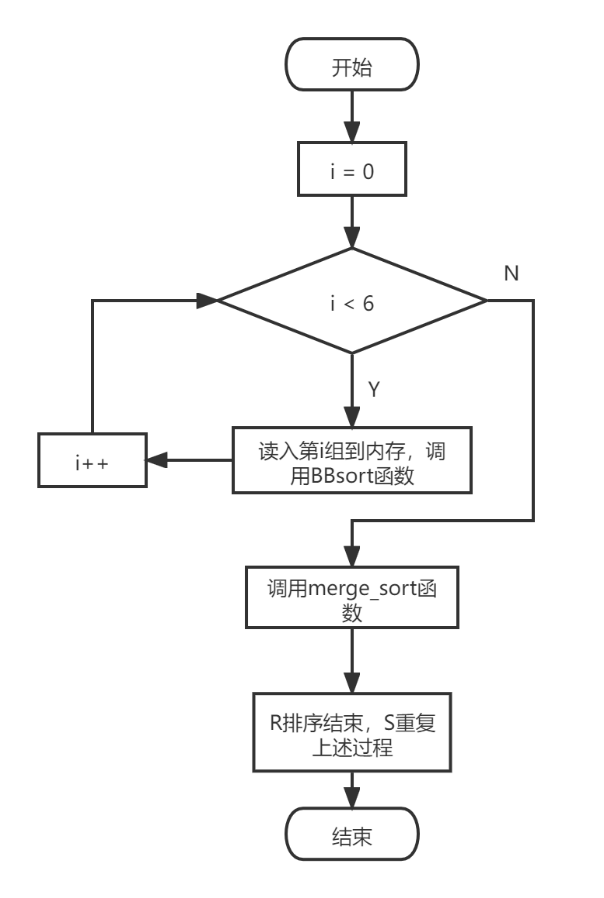
BBsort函数流程图

对于R和S的排序，均选择在第二趟采用六路归并，即使用内存的六个磁盘块进行读，一个磁盘块输出，一个磁盘块作为中间块，便于实现排序时的元组交换。同样，对该过程进行封装为函数merge\_sort()，可以大幅度减少代码量。其流程图如下：



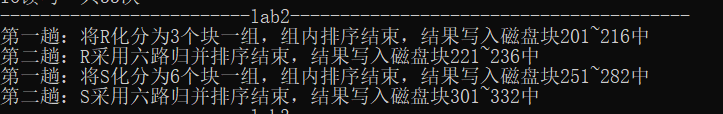
merge\_sort函数流程图

整体流程图：



题目2流程图

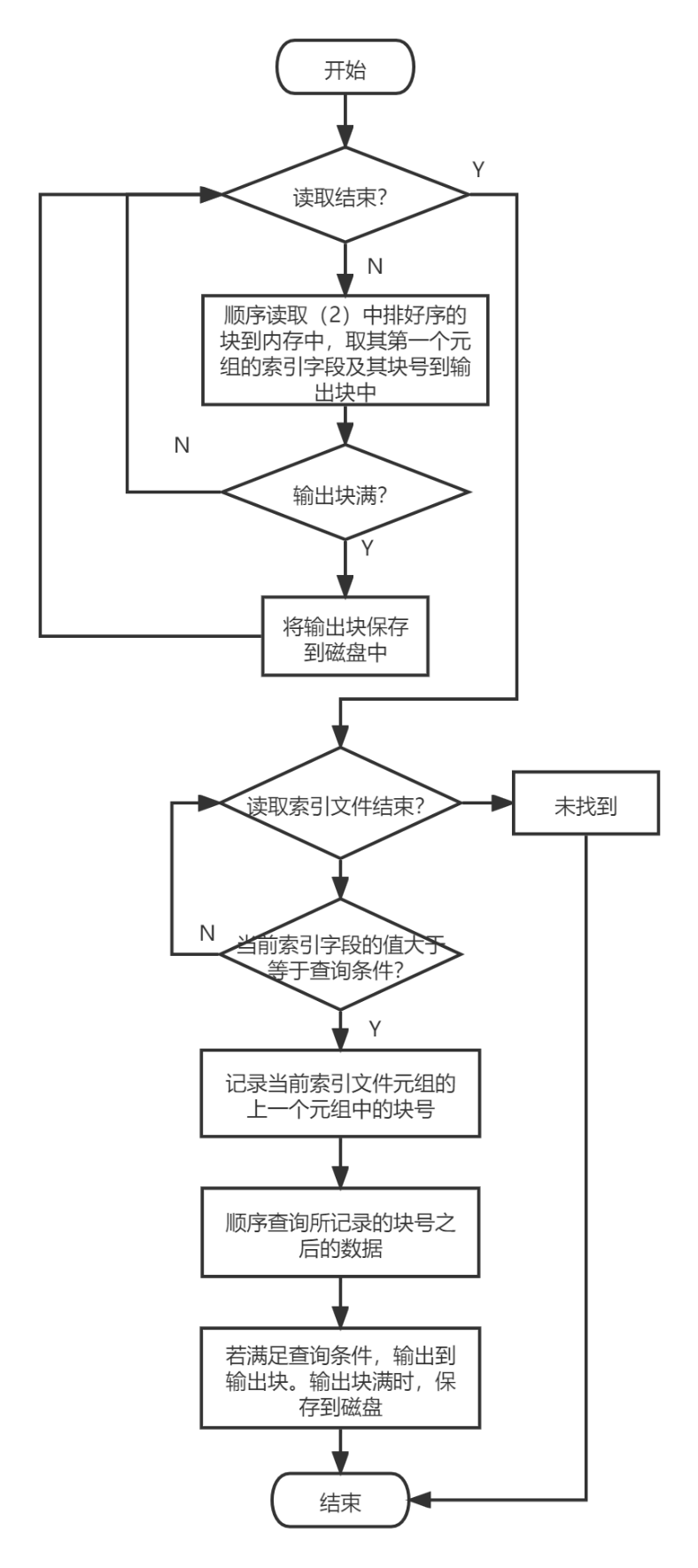
实验结果：



1. **实现基于索引的关系选择算法**

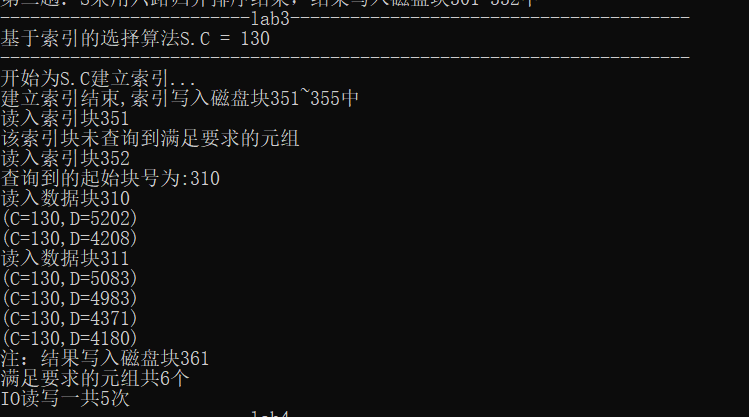
问题分析：首先，对（2）中排好序的文件建立索引，读取每一块的第一个元素，建立主索引文件，并写回到磁盘中。其次，根据选择条件，在索引文件中查询该查询条件中的索引字段处于的磁盘块号，记录下来。最后，根据记录的磁盘块号，读取磁盘数据到内存中，顺序查询满足查询条件的元组，最终写入输出块，并保存到磁盘即可。

流程图：



题目3流程图

实验结果：



1. **实现基于排序的连接操作算法（Sort-Merge-Join）**

问题分析：利用（2）中已经排好序的结果，分别从R、S中读取数据块到内存块blk1,blk2中，并设置指针counter1,counter2分别指向R、S的第一个元组，做如下循环：取当前指针指向的R中元组的连接属性值value1和指向S中元组的连接属性值value2。此时有两种情况：

①当value1和value2不相等时取value1和value2中较小者对应的指针向后移动（封装为getNext()函数），继续循环。

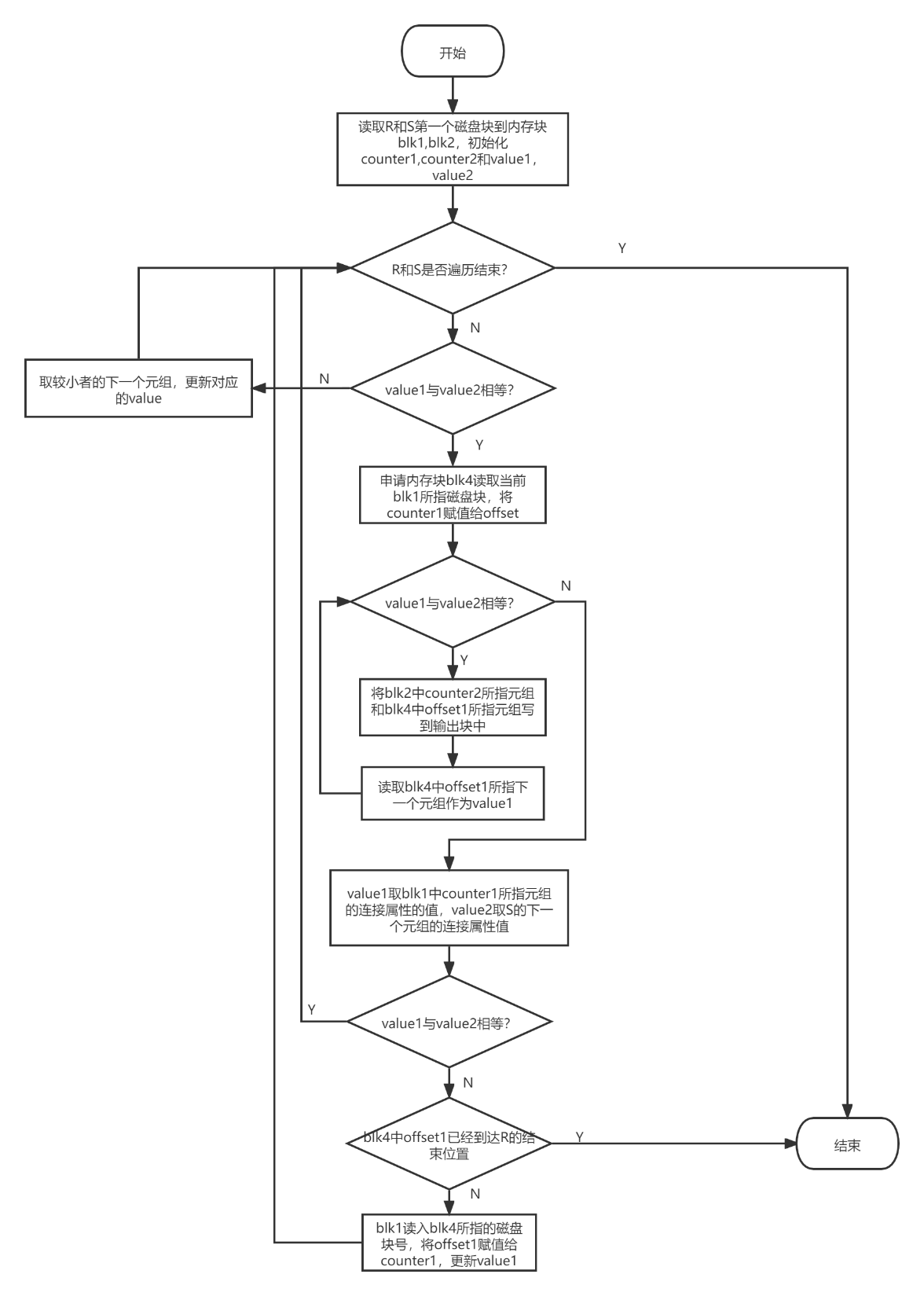
②若value1和value2相等，则满足连接条件，此时，令S的指针counter2不动，申请新的内存块blk4读入当前R所在的磁盘块，用新的指针offset1指向counter1所指的元组，当value1和value2相等时执行下列循环：offset1向后移动，取当前元组的连接属性值为新的value1，进行下一次循环判断;若移动指到当前内存块blk4的末尾且R并未读完，则读入下一块，取新的value1进行下一次循环判断。若R已经读完，则置value1为-1并结束循环。循环结束后，置value1的值为blk1中counter1所指的元组连接属性值。取S的下一个元组连接属性值作为value2，判断此时value2和value1的值：

(i).若不相等，则说明从blk1中counter1所指的元组到刚刚结束循环的blk4所指的磁盘块中第offset1个元组之间的所有元组均与S的下一个元组不匹配，此时可以重新读取blk4所指磁盘块到blk1中，将counter1赋值为offset1，若此时并未读取到R结束，则进行下一次外循环，否则结束外循环。

(ii).相等，则进行下一次外循环

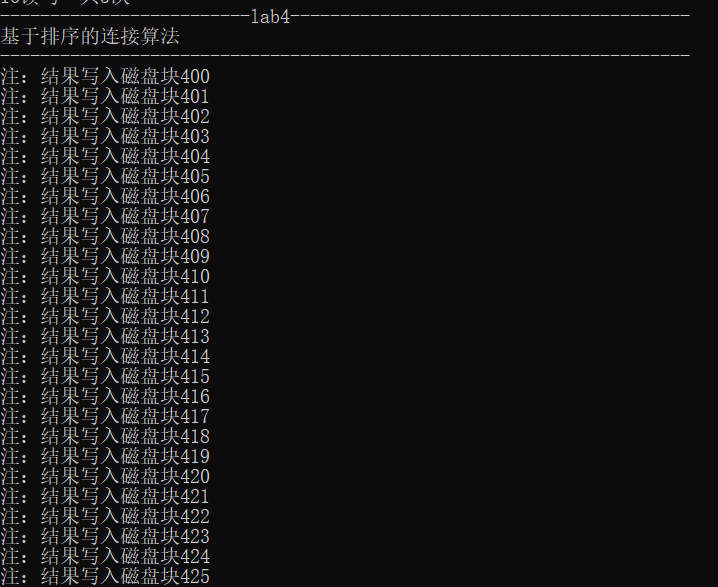
重复上述外循环至有一个关系首先遍历完则结束。

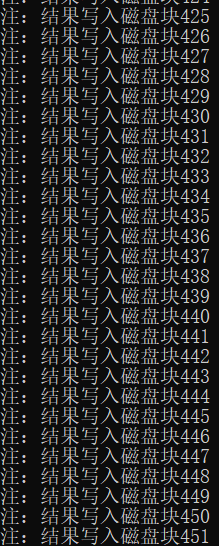
流程图：

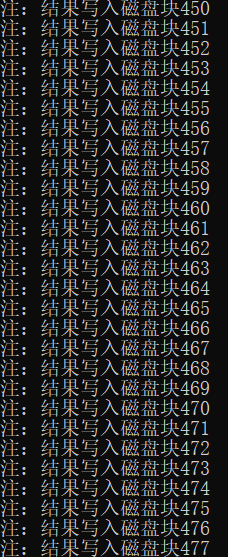


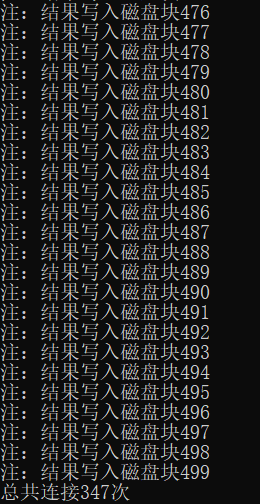
题目4流程图

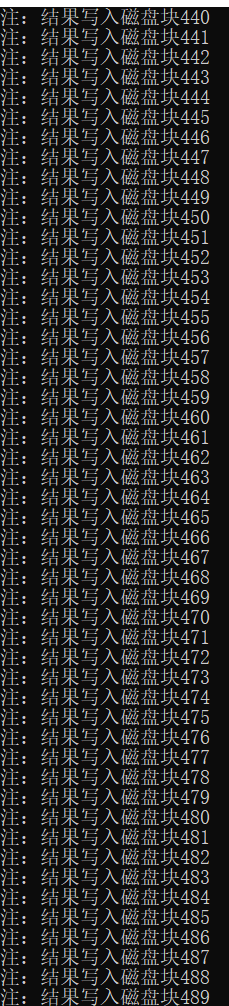
实验结果：

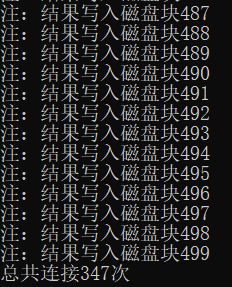












1. **实现基于散列的两趟扫描算法，实现交、并、差其中一种集合操作算法**

问题分析：此问题与（4）基本一样。

利用（2）中已经排好序的结果，分别从R、S中读取数据块到内存块blk1,blk2中，并设置指针counter1,counter2分别指向R、S的第一个元组，做如下循环：取当前指针指向的R中元组的第一个属性值为value1和指向S中元组的第一个属性值为value2。此时有两种情况：

①当value1和value2不相等时取value1和value2中较小者对应的指针向后移动（封装为getNext()函数），继续循环。

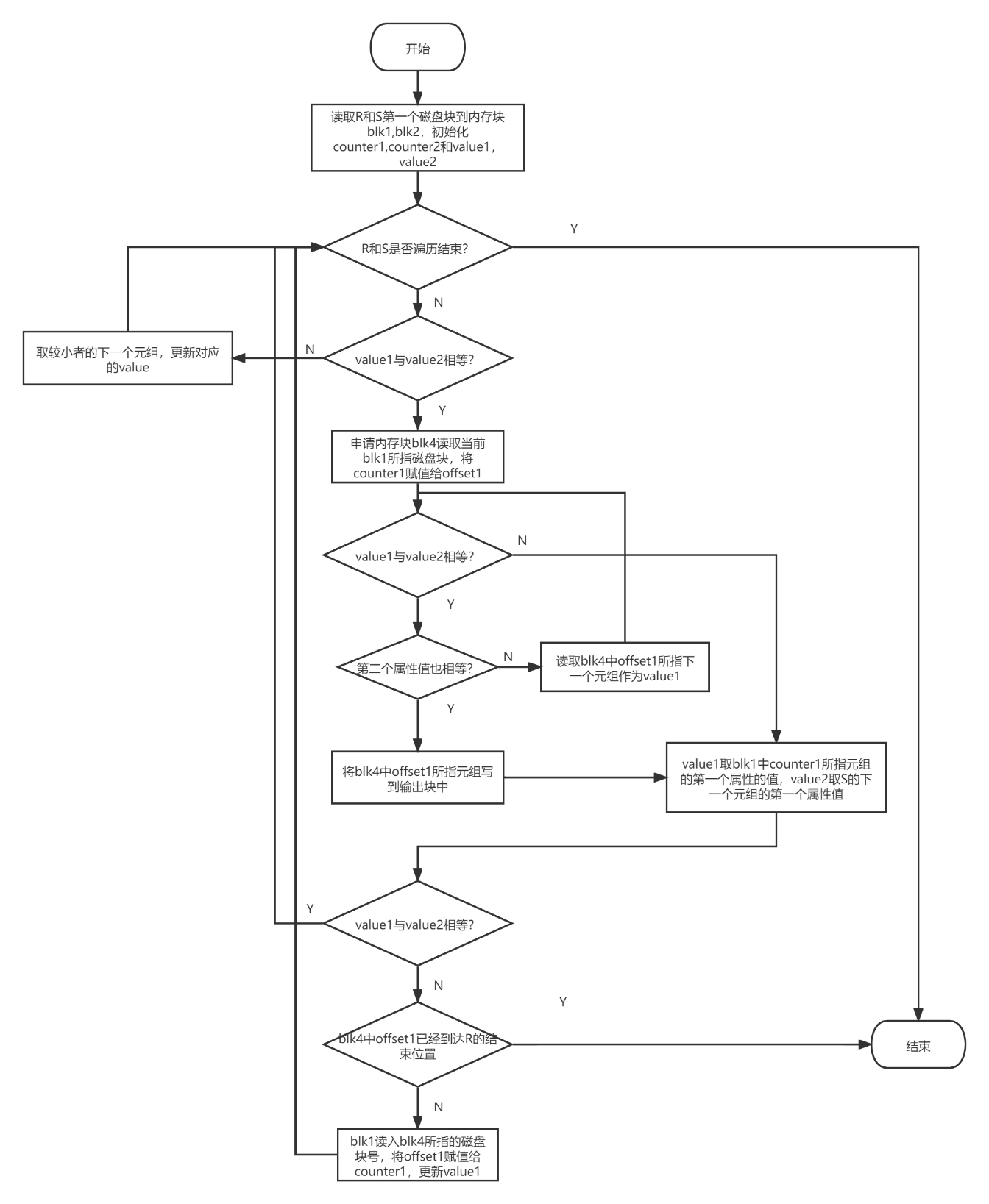
②若value1和value2相等，则满足交运算的第一个条件。此时，令S的指针counter2不动，申请新的内存块blk4读入当前R所在的磁盘块，用新的指针offset1指向counter1所指的元组，当value1和value2相等时执行下列循环：判断当前blk4中offset所指元组和blk2中所指元组的第二个属性值是否相同，若相同则写入输出块并结束循环。否则offset1向后移动，取当前元组的第一个属性值为新的value1，进行下一次循环判断;若移动指到当前内存块blk4的末尾且R并未读完，则读入下一块，取新的value1进行下一次循环判断。若R已经读完，则置value1为-1并结束循环。循环结束后，置value1的值为blk1中counter1所指的元组第一个属性值。取S的下一个元组的第一个属性值作为value2，判断此时value2和value1的值：

(i).若不相等，则说明从blk1中counter1所指的元组到刚刚结束循环的blk4所指的磁盘块中第offset1个元组之间的所有元组均与S的下一个元组不匹配，此时可以重新读取blk4所指磁盘块到blk1中，将counter1赋值为offset，若此时并未读取到R结束，则进行下一次外循环，否则结束外循环。

(ii).若相等，则进行下一次外循环

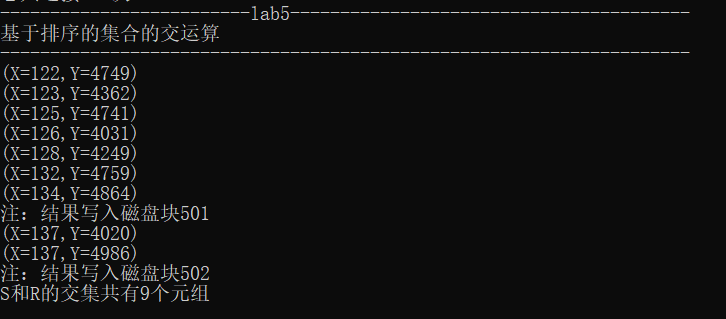
重复上述外循环至有一个关系首先遍历完则结束。

流程图：



题目5流程图

实验结果：



# 总结

遇到的问题：

①由于5个小实验用的是一个内存变量buffer，若不注意申请内存块和释放内存块的同步，则会导致buffer的数据结构维护不好，程序异常的情况。故在每次从磁盘块读取到内存使用完之后，或者在申请一个内存块使用结束之后，要及时释放掉。

②在进行组内排序时，每个块之间还有一个字节的标记位，需要跳过，即偏移量应该多加一个。

收获：

理解了实验三中为经常查询到的字段添加索引的作用，利用索引进行查询能够有效的降低I/O读写次数。加深了对数据库I/O操作算法的理解，本次实验利用有限的内存对存储在磁盘上的关系进行了排序、选择、连接等操作，巩固了我理论课上学习到的算法知识，了解了数据库底层的一些简单算法。

反思：

现在的硬件技术发展到了一定水平，我们使用的计算机内存均比过去增加很多，但是要想存储更大规模的数据，内存仍然不够用，在以后的编程中，仍要考虑内存不足的情况，设计更精妙的算法从而降低I/O次数，从而提高效率。