|  |
| --- |
| 哈尔滨工业大学(深圳) |
| **《数据库》实验报告** |
|  |
| **实验四**  **查询处理算法的模拟实现**  学 院: 计算机科学与技术   |  |  | | --- | --- | | 姓 名: | 林清泉 | | 学 号: | SZ170110314 | | 专 业: | 计算机科学与技术 | | 日 期: | 2020-05-23 | |

# 实验目的

理解索引的作用，掌握关系选择、连接、集合的交、并、差等操作的是实现算法，理解算法的I/O复杂性。

# 实验环境

Windows 10操作系统、CodeBlocks

**注意：CodeBlocks打开方式要选择UTF-8编码，控制台输出时windows 963（默认）编码**

# 实验内容

关系R具有两个属性A和B，其中A和B的属性值均为int型（4个字节），A的值域为[1, 40]，B的值域为[1, 1000]。

关系S具有两个属性C和D，其中C和D的属性值均为int型（4个字节）。C的值域为[20, 60]，D的值域为[1, 1000]。

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R.A=30或S.C=23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R.A=30或S.C=23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. 实现关系投影算法：对关系R上的A属性（非主码）进行投影并去重，将结果存放在磁盘上。
5. 实现基于排序的连接操作算法（Sort-Merge-Join）：对关系R和S计算R.A连接S.C，并统计连接次数，将连接结果存放在磁盘上。
6. 基于排序或散列的两趟扫描算法，实现集合操作算法：并、交、差（S-R）中的任意一种。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**

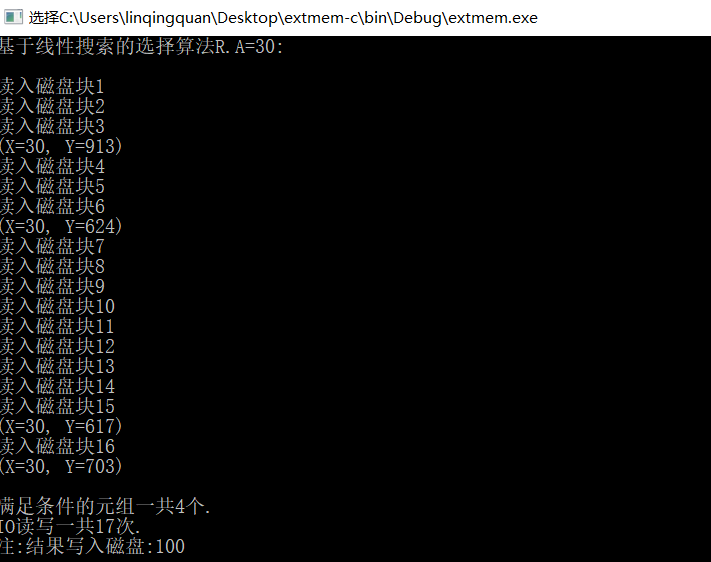
问题分析：基于ExtMem程序库，使用C语言实现线性搜索算法，选出R.A=30的元组，记录I/O读写次数，并将选择结果存放在磁盘上。

内存块一次最多只能读入8个块，但是我们要留一块作为I/O输出，所以，我们一次性最多只能装入7个块，然后将满足结果的元组装入I/O块中进行输出。

读入R关系中的1-16块，注意每次最多只能装7块，将每块中的元组依次取出，并转换为int，若满足条件则输出，并保存至I/O块中，若I/O块已满，则将该块内容写入磁盘，再次新建一个新的I/O块。

搜索结果写入磁盘块：100blk

实验结果：



1. **实现两阶段多路归并排序算法（TPMMS）**

问题分析：实现两阶段多路归并排序算法（TPMMS），因为内存缓冲区一次性不能装入关系R中的所有磁盘块，所以我们用进行两阶段多路归并。首先，对R/S进行分组，我选择分为四个子集合。第一趟，对每个子集合进行排序（我采用的内排序时冒泡排序，跨块根据地址偏移来进行冒泡），对R排序后的四个子集合分别为110-113blk,114-117blk,118-121blk,122-125blk。第二趟，对这四个子集合利用内存缓冲区进行四路归并排序，每次从子集合中各取一块至缓冲区（如果子集合未被取空的话）。R最终归并排序的结果：300-315blk。

对S排序后的四个子集合分别为130-137blk，138-145blk,146-153blk,154-161blk。对S最终归并排序的结果：400-431blk。

实验结果：



1. **实现基于索引的关系选择算法**

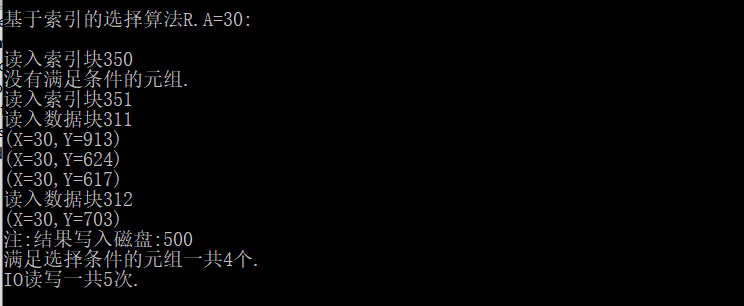
问题分析：利用TPMMS中的排序结果为关系R或S分别建立索引文件，利用索引文件选出R.A=30或S.C=23的元组，并将选择结果存放在磁盘上。记录IO读写次数，与（1）中的结果对比。

先为排好序的R关系建立索引文件（R的索引文件为350-352blk），我采用每个排好序的每个磁盘块中的最大值当做键值索引。索引块中的元组形式为（键值，该值对应的磁盘块号）。然后进行索引查找的时候，先访问索引块，可以加快查询效率。

**难点**：实现索引查找的时候，索引块比较容易建，但如何精准的找到第一个小于等于30并且下一个大于等于30的元组不好找。

搜索结果写入磁盘块500blk。

实验结果：



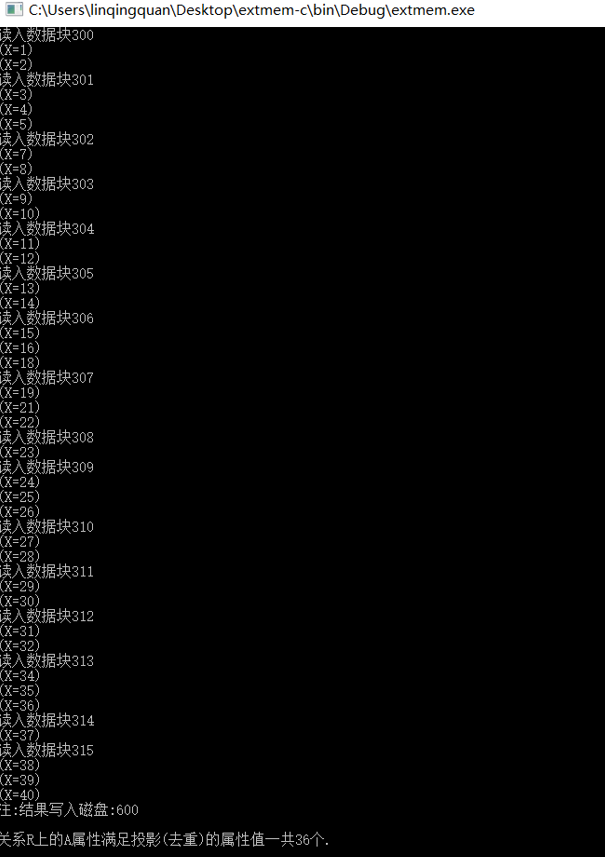
与（1）中线性查找对比，发现查询效率显著提高。

1. **实现关系投影算法**

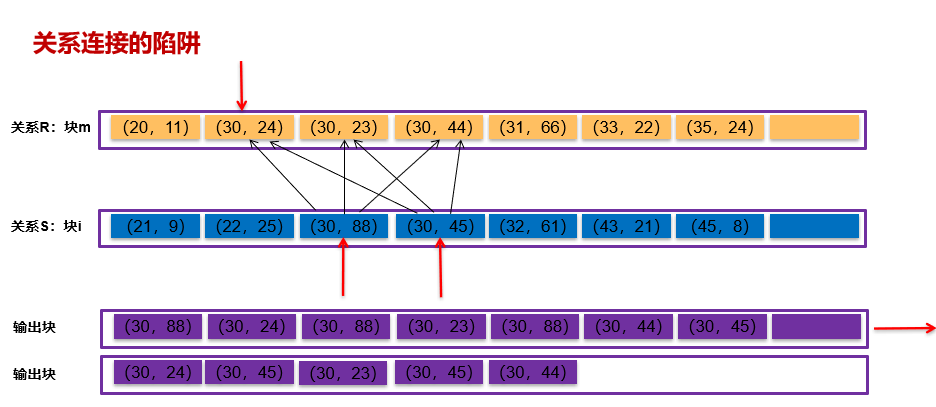
问题分析：对关系R上的A属性（非主码）进行投影并去重，将结果存放在磁盘上。对关系R根据A属性排序，然后根据排序后的文件进行投影并去重

实验结果：600-602blk。

实验结果：

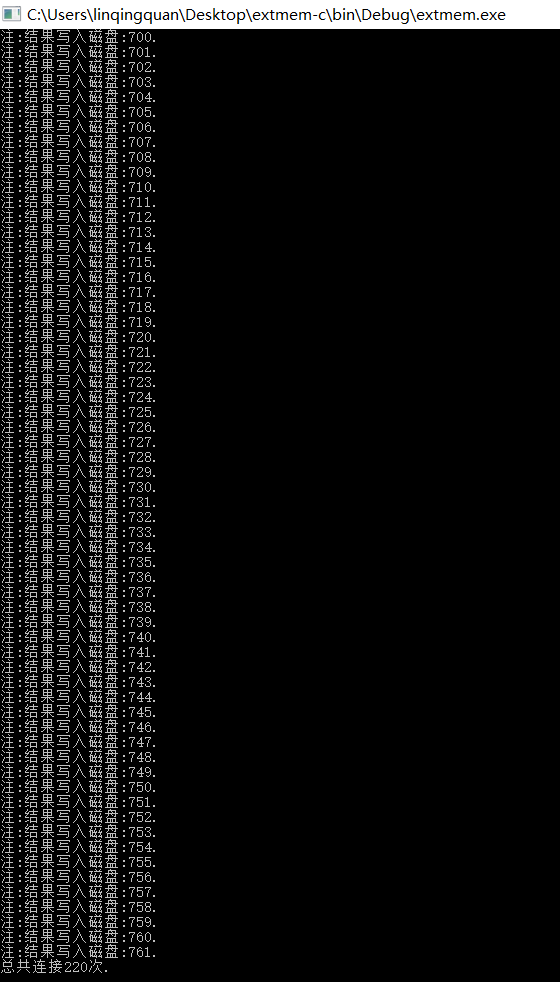


1. **实现基于排序的连接操作算法（Sort-Merge-Join）**

问题分析：对关系R和S计算R.A连接S.C，并统计连接次数，将连接结果存放在磁盘上。本人认为实现关系连接时这六个功能点中最难的一个功能点。

**难点**：如果碰到R.A和S.C相同的元组，需要一个指针not\_move1指向R，一个指针not\_move2指向S，然后not\_move1不动，not\_move2往下挪（有可能跨块），直到R.A不等于S.C，然后not\_move1往下挪一块，not\_move2跳回到最开始相等的位置，实现起来相当的麻烦，我在这块调试了比较久才调试出来。

实验结果：

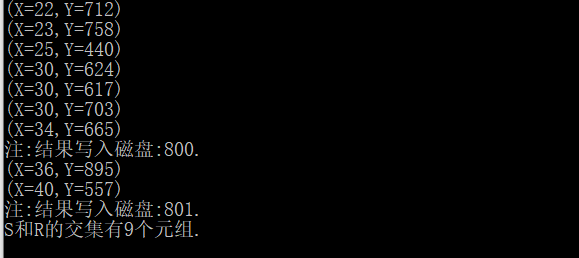


1. **实现集合操作算法**

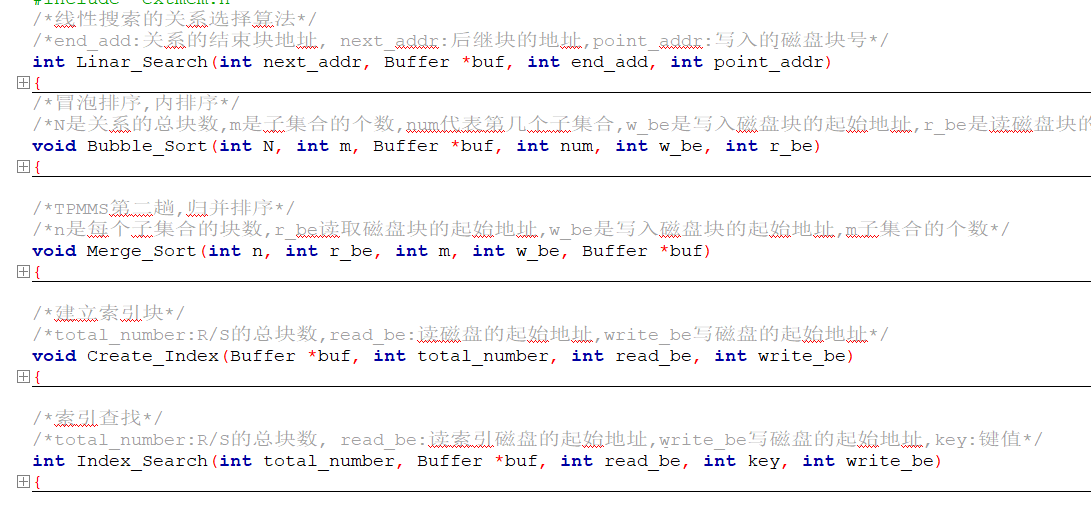
问题分析：我选择实现集合R和S的交操作。该操作和连接操作类似。分别对R和S根据R.A和S.C进行排序，然后根据关系连接的操作进行集合的交操作。**难点：**注意此处的陷阱也和关系连接的类似。当碰到R.A=S.C并且R.B=S.D的时候，也得进行循环，R的指针不动，S的指针往下挪，当R.A=S.C并且R.B=S.D条件不满足的时候，R的指针往下挪一块，S的指针跳回至最开始相等的位置。我最开始就没考虑到这个陷阱，导致相同的元组一直只算出来7个。

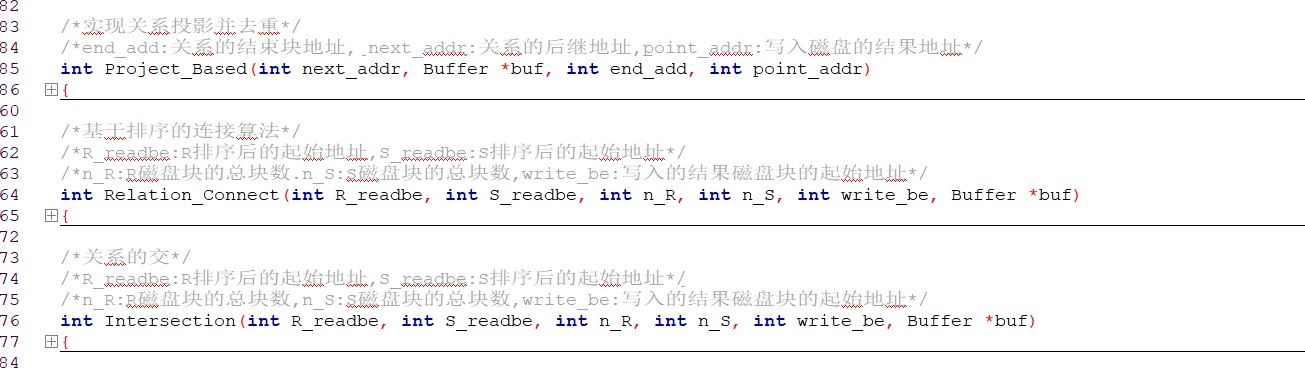
实验结果写入磁盘块800，801blk。

实验结果：



**核心代码讲解：在db\_ex4.c文件中已有对应函数功能的详细解释。**





# 总结

**问题：**

1. 调用int writeBlockToDisk(unsigned char \*blkPtr, unsigned int addr, Buffer \*buf)函数时，这个函数会自动把blk释放掉，导致每次将一块内存区域的内容写至磁盘的时候，之后都得调用这个函数unsigned char \*getNewBlockInBuffer(Buffer \*buf)，重新get一块，这个地方有点坑。
2. 调用void freeBlockInBuffer(unsigned char \*blk, Buffer \*buf)这个函数的时候，也不会自动清洗对应内存区域中的数据。
3. 实现关系连接和交的时候，踩了不少坑。
4. 实现索引查找的时候，索引块比较容易建，但如何精准的找到第一个小于等于30并且下一个大于等于30的元组不好找。

**收获：**

1. 知道了线性搜索和索引搜索的区别，并能够比较好的实现这两部分。
2. 知道了DBMS底层查询算法实现的大体思路。
3. 懂得了TPMMS多路归并排序。
4. 增强了一定的代码能力

**反思：**

1）用API前，如果API不复杂的话，仔细阅读一下API源码会比较好。

2）代码规范有待加强。