|  |
| --- |
| 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: | 邓皓元 | | 学 号: | 200110618 | | 专 业: | 计算机科学与技术 | | 日 期: | 2022-12-31 | |

# 实验目的

*阐述本次实验的目的。*

(1)理解索引、散列的作用

(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

*阐述本次实验的环境。*

Windows 7 操作系统、CodeBlocks

# 实验内容

*阐述本次实验的具体内容。*

1.完成实验任务

实验任务:

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[100, 140]，B 的值域为[400, 500]。

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[120, 160]，D 的值域为[420, 920]。

(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=128 的元组，记录 IO 读写次数，并将选择结果存放在磁盘上。（模拟实现 select S.C, S.D from S where S.C = 128）

(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S建立索引文件，利用索引文件选出 S.C=128 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。（模拟实现 select S.C, S.D from S where S.C = 128）

(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。 （模拟实现 select S.C, S.D, R.A, R.B from S inner join R on S.C = R.A）

(5)实现基于排序或散列的两趟扫描算法，实现并（S ∪R）、交（S ∩R）、差（S - R）其中一种集合操作算法，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

*对实验中的5个题目分别进行分析，并对核心代码和算法流程进行讲解，用自然语言描述解决问题的方案 。并给出程序正确运行的结果截图。*

1. **实现基于线性搜索的关系选择算法**

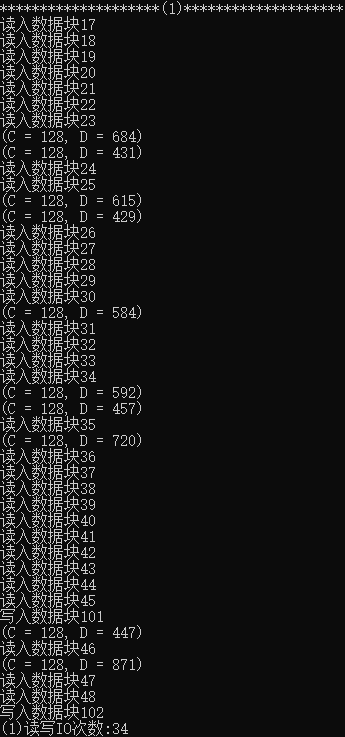
问题分析：

即是一个普通线性遍历搜索算法的实现，用一个用以遍历的blk\_get数据块遍历整个S关系，检测S关系的每个元组，检测到元组C属性为128则将元组加入到用以记录的blk\_write数据块中，

其中blk\_get数据块每次访问一个64字节大小的blk文件，检测其中的7个元组，符合条件就将对应元组存入blk\_write数据块

blk\_write数据块每次存入一个8字节大小的元组，blk\_write数据块满了则按顺序写入磁盘中，将blk\_write初始化为一个新数据块，遍历完毕之后将当前blk\_write按顺序写入磁盘中。

实验结果：



1. **实现两阶段多路归并排序算法（TPMMS）**

问题分析：

将R(S)关系总集合按照一定大小划分成N个子集合，子集合块数需要小于内存缓冲区可用块数，一次将一个子集合完全存入内存缓冲区中，排序好之后存入磁盘，类似地将N个子集合的排序结果存入磁盘后。

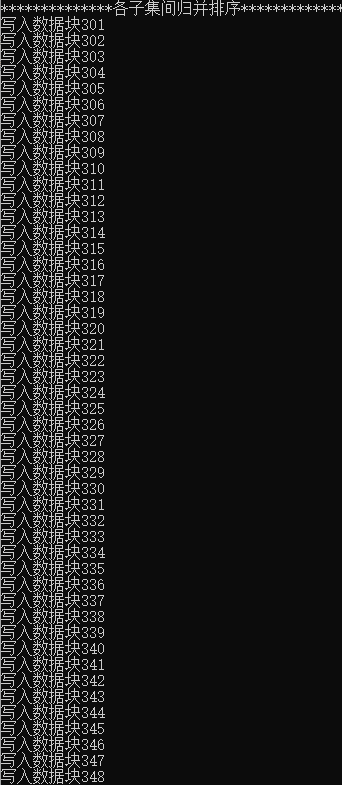
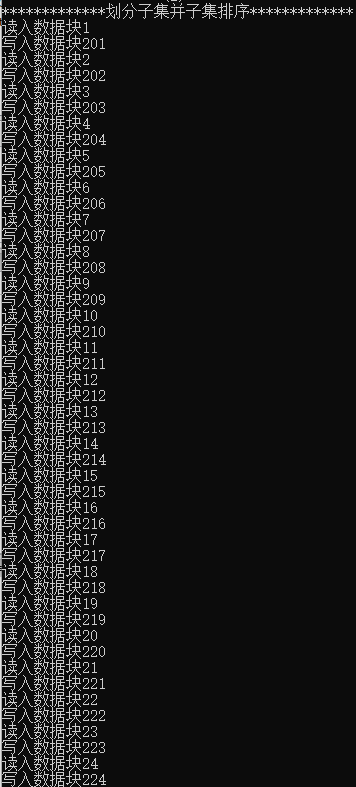
假设子集合按照升序排序，将每个排序子集合的第一个元组存入缓冲区中(8个字节)则缓冲区大小至少得8N+64字节(额外一个块用以写磁盘)，每次找到缓冲区内这些元组(A或者C)最小的元组，取出，按顺序放到用以写磁盘的数据块中，在被取出元组的原来的排序子集合中取出第二个元组存到当前位置。

若是排序子集合中元组都被取完了，则存入一个最大的元组来避免该元组被提取(999,999)，一直到所有元组都被取出(取固定次数例如R16\*7次S32\*7次)。

用以写磁盘的数据块每次被存满了则将数据块内数据写入磁盘，初始化为新的空数据块。

实验结果：排序好的子集合数据块写入201-248位置上

排序总结果写入301-348位置上



1. **实现基于索引的关系选择算法**

问题分析：

将(2)排序好的集合的每个数据块的第一个元组和每个数据块的地址作为一个索引(数据块第一个元组属性C或者属性A数据4字节+数据块地址4字节)8字节，

则R关系需要16个索引，16\*8=128字节=2数据块作为索引块

将351-352作为R关系关于属性A的索引块

则S关系需要32个索引，32\*8=256字节=4数据块作为索引块

将353-356作为S关系关于属性C的索引块

在检测关系S的属性C为128值时，利用索引块得到排序好的数据块每个数据块的第一个元组的值，首先找到元组值为128之前的第一个数据块地址beginaddr，再找到值为128的最后一个数据块地址endaddr，若是beginaddr不存在则直接取317(关系S第一个排序集合)，若是endaddr不存在则直接取348(关系S最后一个排序集合)，最后遍历访问beginaddr-endaddr数据块，找到其中属性C为128的元组并存入磁盘，这里得到的beginaddr=323，endaddr=325

实验结果：

首先利用(2)的排序结果创建索引块



R的2个索引块：

351索引块：



352索引块：



S的4个索引块：

353索引块：



354索引块：

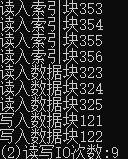


355索引块：



356索引块：





数据块121中的数据：



数据块122中的数据：



经检验和(1)线性搜索结果一样，

其中IO读写次数为9，远远小于(1)IO读写次数34

1. **实现基于排序的连接操作算法（Sort-Merge-Join）**

问题分析：

利用(2)排序好的集合进行关系R和关系S的连接操作，由于要实现R属性A和S属性C的连接，A属于[100,140]，C[120,160]。

因此首先遍历关系R的属性A范围为120-140的元组(可以利用索引)利用一个blk\_A数据块，对关系R的每个元组遍历S属性B范围为120-140的元组(可用利用索引)利用一个blk\_C数据块，若是属性B的值等于属性A的值则将该两个元组加入用以写入磁盘的数据块中(S.C，S.D，R.A，R.B)

实验结果：R和S连接集389个



1. **实现基于排序的两趟扫描算法，实现并集集合操作算法**

问题分析：

利用(2)排序好的集合进行关系R和关系S的并操作。

首先遍历关系S所有元组，都存入用以写入磁盘的数据块中，之后遍历关系R所有元组，对关系R中的每个元组，遍历关系S所有元组，看看是否能够找到和R相等的元组(R.A == S.C && R.B == S.D)，若是找到符合条件的元组则不存入用以写入磁盘的数据块中，否则存入用以写入磁盘的数据块中

实验结果：



# 附加题

*对剩余的两种集合操作进行问题分析，并给出程序正确运行的结果截图。*

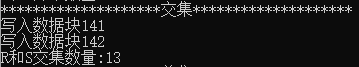
1. **实现基于排序的两趟扫描算法，实现交集集合操作算法**

问题分析：

利用(2)排序好的集合进行关系R和关系S的交操作。

首先遍历关系S所有元组，对关系S中的每个元组，遍历关系R所有元组，看看是否能够找到和R相等的元组(R.A == S.C && R.B == S.D)，若是找到符合条件的元组则存入用以写入磁盘的数据块中，否则不存入用以写入磁盘的数据块中

实验结果：

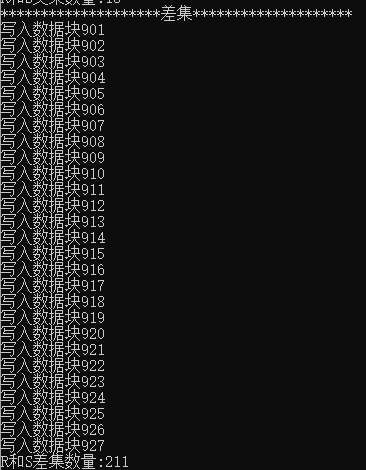


1. **实现基于排序的两趟扫描算法，实现差集集合操作算法**

问题分析：

首先遍历关系S所有元组，对关系S中的每个元组，遍历关系R所有元组，看看是否能够找到和R相等的元组(R.A == S.C && R.B == S.D)，若是找到符合条件的元组则不存入用以写入磁盘的数据块中，否则存入用以写入磁盘的数据块中

实验结果：



# 总结

*总结本次实验的遇到并解决的问题、收获及反思。*

在本次实验中，在某些数据块的存入磁盘情况下遇到了问题，我向一块初始化的数据块中写入较少数据，在该数据块存入磁盘的时候却存入了较多数据，为了了解错误的产生我去查看了数据块存取和初始化的源码，并发现较快地进行已经写入的数据块的初始化会导致该数据块中仍旧存有先前存放的数据，因此我改写了将数据块写入磁盘的代码，增加了一个size参数用来传入数据块需要写入磁盘内数据的大小来判断需要写多少数据入磁盘，这让我明白彻底了解代码底层运行逻辑对于问题的解决是至关重要的最佳捷径。