|  |
| --- |
| 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: | 孙铎 | | 学 号: | 200110503 | | 专 业: | 计算机科学与技术 | | 日 期: | 2023-01-01 | |

# 实验目的

*阐述本次实验的目的。*

* 理解索引、散列的作用；
* 掌握关系选择、投影、连接、集合的交、并、差等操作的实现算法；
* 加深对算法I/O复杂性的理解；
* 理解两阶段多路归并排序算法的思想、实现与应用；
* 理解、实现并运用简单的索引。

# 实验环境

*阐述本次实验的环境。*

* Windows 10 操作系统
* gcc version 8.1.0 (x86\_64-win32-seh-rev0, Built by MinGW-W64 project)
* CodeBlocks

# 实验内容

*阐述本次实验的具体内容。*

* 基于ExtMem程序库，模拟实现数据库的一些查询处理算法；
* 实现关系选择、连接操作算法；
* 实现简单的索引，并实现基于索引的关系选择算法；
* 实现集合并、交、差操作算法。

# 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. **对extmem.c的修改以及utils.c工具函数的封装**

* 修改extmem.c：
  + 修改freeBlockInBuffer函数：freeBlockInBuffer函数原本只将内存块标志位设置为可用，并没有真正清空内存块数据，这里将其修改为，**调用memset将内存块全部置0**并设置**标志位**为可用；
  + 修改writeBlockToDisk函数：将writeBlockToDisk函数中将内存块标志位设置为可用的代码修改为**对freeBlockInBuffer函数的调用**，这样实现的效果是一致的。
* 封装utils.c中的工具函数：
  + record2XY函数：
    - 输入：
      * blk：内存块指针；
      * recordNum：要**读出**的块中的记录；
      * X：接收该条记录的**前4个字节**的传出参数；
      * Y：接收该条记录的**后4个字节**的传出参数。
    - 输出：无
    - 函数实现：
      1. 声明字符数组str，长度为5；
      2. 通过recordNum进行下标运算，读取blk对应记录的**前4个字节**到str中，并调用**atoi函数**将str转换成int型整数赋值给传出参数X；
      3. 通过recordNum进行下标运算，读取blk对应记录的**后4个字节**到str中，并调用**atoi函数**将str转换成int型整数赋值给传出参数Y。
  + XY2record函数：
    - 输入：
      * blk：内存块指针；
      * recordNum：要**写入**的块中的记录；
      * X：向记录**前4个字节**写入的数值；
      * Y：向记录**后4个字节**写入的数值。
    - 输出：无
    - 函数实现：
      1. 声明字符数组fourBytes，长度为4；
      2. 判断X**是否等于-1**，如果不等于-1，则**调用sprintf将X转换为字符串**存储至fourBytes中，并通过recordNum进行下标运算，将fourBytes中的4个字节写入blk对应记录的**前4个字节**，如果等于-1，则将**数值0**写入blk对应记录的**前4个字节**；
      3. 判断Y**是否等于-1**，如果不等于-1，则**调用sprintf将Y转换为字符串**存储至fourBytes中，并通过recordNum进行下标运算，将fourBytes中的4个字节写入blk对应记录的**后4个字节**，如果等于-1，则将**数值0**写入blk对应记录的**后4个字节**。
  + nextAddr函数
    - 输入：
      * blk：内存块指针
    - 输出：blk块的后继块地址值
    - 函数实现：
      1. 声明字符数组str，长度为5；
      2. 通过下标运算，读取blk**最后8个字节的前4个字节**到str中，**调用atoi函数**将str转换成int型整值，并返回该值（约定将**后继块地址**存储在blk**最后8个字节的前4个字节**中）。
  + shiftRecord函数
  + writeToOutBlk函数

1. **实现基于线性搜索的关系选择算法**

问题分析：

* 要将关系所在的磁盘块依次**读入内存**，并要依次判断各条记录**是否符合选择条件**，并**输出符合选择条件的记录**到磁盘块上。因此，需要重点实现的操作是：
  + 遍历读入的磁盘块中各条记录，**正确解析其数据值**，并判断是否符合选择条件；
  + 将符合选择条件的记录**暂存至新的内存块**，待内存块满时**输出至磁盘块**，同时也要设置好其**后继地址**。
* 核心代码与算法流程：
  1. 定义变量：
     + 1个**内存缓冲区buf**；
     + 1个**内存块指针blk**用于存放**关系S的数据块**；
     + 1个**内存块指针resBlk**用于暂存满足选择条件**待输出的记录**；
     + 1个**记数变量rowCount**记录满足选择条件的**记录条数**。
  2. 调用**initBuffer函数初始化buf**为总大小520B，块大小64B的内存缓冲区；
  3. 从**起始磁盘块17**开始，通过解析其后继地址，**遍历关系S的所有块**，并在每次遍历时执行如下操作：
     1. 调用**readBlockFromDisk函数读取**遍历到的磁盘块至内存块，其指针存放至blk中；
     2. 遍历块中的**前7条记录**，调用record2XY函数将每条记录的**S.C和S.D**分别存入**变量X和变量Y**中；
     3. 如果**X等于选择条件128**，则输出X和Y的值，令**rowCount加1**，并判断resBlk中之前是否已经**写满了7条记录**，如果已经写满，则调用XY2record函数，将**后继块地址**写入resBlk的最后8个字节，并调用writeBlockToDisk函数**将resBlk输出至对应磁盘位置**，再调用getNewBlockInBuffer函数**重新获取空闲块**给resBlk，无论resBlk是否已经写满，最后都要再调用XY2record函数，**将当前的X和Y值写入**resBlk的下一条空记录中；
     4. 调用nextAddr函数读取**后继块地址**，调用freeBlockInBuffer函数释放blk，继续遍历关系S的下一块。
  4. 将**最后一个不超过7条选择结果记录**的resBlk写入磁盘。因为之前每次都是写满一块后，**下一次要写入新的块时才将上一块写入磁盘**，所以遍历结束后，只要存在满足选择条件的记录，resBlk中就一定有**剩余待输出的选择结果记录**；
  5. 输出一些结果信息，详见下方的**实验结果**。
  6. 调用freeBuffer函数**释放内存缓冲区buf**

实验结果：

文本

描述已自动生成

1. **实现两阶段多路归并排序算法（TPMMS）**

问题分析：

* 首先需要将磁盘块**分组读入内存**，进行**内排序**，输出**中间结果**至磁盘，再将磁盘上的内排序结果按照**归并排序的方式读入内存处理**，并将归并结果**输出至磁盘**，最后**擦除内排序中间结果**。因此，需要重点实现的操作是：
  + 内排序：
    - 将磁盘块**分组读入内存**；
    - 对每组磁盘块使用内排序算法**排序所有记录**；
    - 将内排序中间结果分块输出至磁盘。
  + 归并排序：
    - 为每组磁盘块安排好内存块，**每组分别读入一部分到内存**；
    - **横向比较每组内存块的记录**，按排序方式每次选择一条记录输出；
    - 当一组内存块的记录全部处理完时，需要**换入该组的下一块**，当该组全部块都处理完时，需要对其进行标记。
* 核心代码与算法流程：
  1. 定义1个内存缓冲区buf，调用**initBuffer函数初始化buf**为总大小520B，块大小64B的内存缓冲区；
  2. 调用封装好的**函数internalSort**分别对关系R和关系S分组进行**内排序**，internalSort函数定义如下：
     1. 声明大小为8的内存块指针数组blks，用于存放**关系的一组数据块**；
     2. 从关系存放的起始块开始，通过nextAddr函数获取后继块地址，**循环读入关系的每个数据块**；
     3. 每读入8个数据块，进行一次**冒泡排序**，排序8个数据块中的所有记录，利用冒泡排序的规则，遍历所有记录，调用record2XY函数**将相邻两条记录字段C和字段D的值分别读入**X1、X2和Y1、Y2，并比较，如果X1Y1比X2Y2大，则再调用XY2record函数**交换相邻两条的记录值**，重新写入对应磁盘块的记录上；
     4. 冒泡排序结束后，调用writeBlockToDisk函数，**循环输出该组的每一块至磁盘**，并输出中间结果存储到的磁盘块号，除了关系的最后一个数据块，其他数据块都要调用XY2record函数将**后继地址**写入块的最后8个字节上。
  3. 调用封装好的**函数externalSort**分别对关系R和关系S分组进行**归并排序**，externalSort函数定义如下：
     1. 定义变量：
     + blks：数组，存储**各组读入内存的数据块指针**，**每组分配1个内存块**；
     + recordCnts：数组，存储各**组读到了第几条记录**；
     + blk：**1个**暂存归并排序**输出记录的数据块**；
     + recordCnt：**循环计数变量**，存储下一次应当将结果写到**blk上第几条记录**，从0开始，最大值为6，到达6后循环回到0。
     1. 将**每组的第1个块**先读入内存，其指针分别存储在blks数组中；
     2. 调用getNewBlockInBuffer函数为blk申请可用内存块；
     3. 循环从**每组**对应的内存块中**分别调用record2XY函数读入一条记录**，其中对应的**块指针**存储在blks数组中，对应块上的**记录号**存储在recordCnts数组中，如果某一组已经**处理完所有记录**，会将其recordCnts数组对应元素**标记为-1**（在后面的步骤中有体现），先比较A或C字段，A或C字段相等再比较B或D字段，将**各组最小的记录值**存储到变量Xpre和Ypre中，并将**选择到最小记录的组号**存储在变量select中；
     4. **将最小记录调用XY2record函数写入blk**的recordCnt条记录处，并令recordCnt加1，如果blk已经**写满了7条**，则调用writeBlockToDisk函数将blk**输出至磁盘对应位置**，并再次调用getNewBlockInBuffer函数为blk申请可用内存块，同时，如果不是最后一块，则还要调用XY2record函数向块中写入**后继地址**（因为已知关系R和关系S的记录总数为7的倍数，所以这里简化处理，只在写满7条时输出，也能保证输出关系的所有记录没有剩余）；
     5. 令recordCnts[select]加1，即本次被选择的组要**后移一条记录**，如果该组已经处理完**最后一条记录**，则将recordCnts[select]**标记为-1**，如果不是最后一条记录，但通过recordCnts[select]判断出来已经**处理完了一整个块**，则调用nextAddr函数读出**后继地址**，并调用freeBlockInBuffer函数释放blks[select]，然后调用readBlockFromDisk函数读入后继块到blks[select]中，回到步骤iv.继续参与循环处理，直到所有组的所有记录均处理完。
  4. 循环调用dropBlockOnDisk函数，**擦除内排序中间结果**；
  5. 调用freeBuffer函数**释放内存缓冲区buf**。

实验结果：

图片包含 文本

描述已自动生成

……

文本

描述已自动生成

1. **实现基于索引的关系选择算法**

问题分析：

* 首先要**设计索引块结构**，为排好序的关系S**建立索引**，然后**遍历各索引块**，根据**索引项指针**快速定位满足条件的记录应该出现在哪些块中，然后遍历这些块中的记录，**选择满足筛选条件的记录**并输出。因此，需要重点实现的操作是：
  + 遍历排好序的关系S，为**每一块建立一条索引项存放在索引块**中，并输出索引块；
  + 遍历索引块，对比**索引项上的值和筛选条件**，通过**索引项上的指针**快速定位满足条件记录所在块，选择记录并输出。
* 核心代码与算法流程：
  1. 调用buildIndex函数，**为S.C建立索引**，返回**索引块的起始地址**，其中buildIndex函数的定义如下：
     1. 定义变量：
     + dataBlk：存储读入内存的关系S的**1个数据块**的指针；
     + idxBlk：存储要输出至磁盘的**1个索引块**指针。
     1. 定义1个内存缓冲区buf，调用**initBuffer函数初始化buf**为总大小520B，块大小64B的内存缓冲区（注：这里之所以单独定义一个内存缓冲区，是因为**后面还要统计基于索引的选择操作的I/O次数**，如果使用同一个缓冲区建立索引，则**I/O次数会叠加**，所以**建立索引时单独使用一个缓冲区**）；
     2. 调用getNewBlockInBuffer函数为idxBlk申请可用内存块；
     3. 通过起始地址、nextAddr函数和readBlockFromDisk函数，读入磁盘块到dataBlk中，**循环遍历排好序的关系S的所有数据块**，每次遍历一个数据块时，调用record2XY函数读取**数据块第一条记录**，用于创建**索引项**，如果索引块中已经写满了7个8字节索引项，则**先输出该索引块**，调用XY2record写入**后继地址**，然后调用writeBlockToDisk函数输出至磁盘，再调用getNewBlockInBuffer函数重新为idxBlk申请空闲内存块，无论索引块是否写满7项，都要再调用XY2record函数将该条记录**C字段值和块地址写入索引块**的8个字节上，最后调用freeBlockInBuffer函数释放dataBlk，待下一次循环重新读入磁盘块到dataBlk中（注：要保证**不同索引项的索引值不重复**，这可以通过记录上一索引项的值，并与当前值进行对比来实现）；
     4. 调用writeBlockToDisk函数**输出最后一个不超过7项的索引块**至磁盘；
     5. 调用freeBuffer函数**释放内存缓冲区buf**。
  2. 定义1个内存缓冲区buf，调用**initBuffer函数初始化buf**为总大小520B，块大小64B的内存缓冲区；
  3. 定义变量：
     + dataBlk：存放读入内存的关系S**数据块**指针；
     + idxBlk：存放读入内存的S.C**索引块**指针；
     + resBlk：暂存待输出至磁盘的筛选记录。
  4. 通过索引块起始地址、nextAddr函数和readBlockFromDisk函数，**读入索引块到indexBlk**中，**顺序遍历各索引块**，每次遍历执行如下操作：
     1. 通过record2XY函数读入索引块中的**各条索引项**，如果当前索引项的值**大于目标值128**，或者当前索引项是**空的**（这说明已经找完了最后一个索引项），说明**从上一个索引项指针的数据块到当前索引项指向的数据块**（如果已经找完了最后一个索引项，则是直接到最后一个数据块）中可能出现与目标值128匹配的记录，则需要回退索引项，遍历对应数据块记录，标记flag为1，退出当前遍历索引项的循环；
     2. 结束索引项循环后，判断flag，如果…..
     3. 如果flag为0，则输出信息“没有满足条件的元组”，并继续遍历下一索引块
  5. 将最后….
  6. 输出信息….
  7. 调用freeBuffer函数**释放内存缓冲区buf**。

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：

* xxxxx
* 核心代码与算法流程：
  1. 定义1个内存缓冲区buf，调用**initBuffer函数初始化buf**为总大小520B，块大小64B的内存缓冲区；
  2. 调用freeBuffer函数**释放内存缓冲区buf**。

实验结果：

1. **实现基于排序的两趟扫描算法，实现交集合操作算法**

问题分析：

* xxxxx
* 核心代码与算法流程：
  1. 定义1个内存缓冲区buf，调用**initBuffer函数初始化buf**为总大小520B，块大小64B的内存缓冲区；
  2. 调用freeBuffer函数**释放内存缓冲区buf**。

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，实现并集合操作算法**

问题分析：

* xxxxx
* 核心代码与算法流程：
  1. 定义1个内存缓冲区buf，调用**initBuffer函数初始化buf**为总大小520B，块大小64B的内存缓冲区；
  2. 调用freeBuffer函数**释放内存缓冲区buf**。

实验结果：

1. **实现基于排序的两趟扫描算法，实现差集合操作算法**

问题分析：

* xxxxx
* 核心代码与算法流程：
  1. 定义1个内存缓冲区buf，调用**initBuffer函数初始化buf**为总大小520B，块大小64B的内存缓冲区；
  2. 调用freeBuffer函数**释放内存缓冲区buf**。

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。*

* 之后可以不用Codeblocks吗，Bug太多了
* 4个字节的存储方式，请说清存的是字符而不是二进制数
* 写个OJ可以不
* 明确要求，明确连续和离散，明确内存块利用率