哈尔滨工业大学（深圳）

# 数据库实验指导书

实验五 查询处理算法的模拟实现

# 目录

# 1. 实验目的

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

# 2. 实验环境

Windows 7 操作系统、CodeBlocks。

# 3. 实验内容

## 3.1 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。

**注意：**

**（1）每人需要完成上述全部算法，根据完成情况给分。**
**（2）本次实验需要提交实验报告和工程文件。**
**（3）要求使用有限内存（ Buffer ）实现上述算法，不可定义长度大于 10 的整型或字符型数组。**

## 3.2 附加题

* 基于排序或散列的两趟扫描算法，实现剩余的两种集合操作算法。 （选做）

请将结果存放在磁盘上，并统计并、交、差操作后的剩余元组个数。

## 3.3 ExtMem 程序库介绍

ExtMem 程序库是一个专门为本课程编写的模拟外存磁盘块存储和存取的程序库，由 C 语言开发。ExtMem 程序库的功能包括内存缓冲区管理、磁盘块读/写，它提供了 1 个数据结构和 7 个 API 函数。

ExtMem 程序库定义了 Buffer 数据类型，包含如下 6 个域：

- numIO：外存 I/O 次数；
- bufSize：缓冲区大小（单位：字节）；
- blkSize：块的大小（单位：字节）；
- numAllBlk：缓冲区内可存放的最多块数；
- numFreeBlk：缓冲区内可用的块数；
- data：缓冲区内存区域。

缓冲区中每个块内数据大小为 blkSize 个字节，其最后 4 个字节用来存放其后继磁盘块的地址（在 ExtMem 库中，我们 4 个字节来记录磁盘块地址，地址在程序中为 unsigned int 类型。若无后继磁盘块，则置为 0），其余(blkSize － 4)个字节用于存放块内的记录。缓冲区每个块**之前有 1 个字节的标志位表示是否被占用**。

ExtMem 库提供了如下 API 函数：

- Buffer *initBuffer(size_t bufSize, size_t blkSize, Buffer *buf);
  初始化缓冲区，其输入参数 bufSize 为缓冲区大小（单位：字节），blkSize 为块的大小（单位：字节），buf 为指向待初始化的缓冲区的指针。若缓冲区初始化成功，则该函数返回指向该缓冲区的地址；否则，返回 NULL。
- void freeBuffer(Buffer *buf);
  释放缓冲区 buf 占用的内存空间。
- unsigned char *getNewBlockInBuffer(Buffer *buf);
  在缓冲区 buf 中申请一个新的块。若申请成功，则返回该块的起始地址；否则，返回 NULL。

- void freeBlockInBuffer(unsigned char *blk, Buffer *buf);
  解除块 blk 对缓冲区内存的占用，即将 blk 占据的内存区域标记为可用。
- int dropBlockOnDisk(unsigned int addr);
  从磁盘上删除地址为 addr 的磁盘块内的数据。若删除成功，则返回 0；否则，返回-1。
- unsigned char *readBlockFromDisk(unsigned int addr, Buffer *buf);
  将磁盘上地址为 addr 的磁盘块读入缓冲区 buf。若读取成功，则返回缓冲区内该块的地址；否则，返回 NULL。同时，缓冲区 buf 的 I/O 次数加 1。
- int writeBlockToDisk(unsigned char *blkPtr, unsigned int addr, Buffer *buf);
  将缓冲区 buf 内的块 blk 写入磁盘上地址为 addr 的磁盘块。若写入成功，则返回 0；否则，返回-1。同时，缓冲区 buf 的 I/O 次数加 1。

**文件 test.c 中给出了 ExtMem 库使用方法的具体示例。**

**声明：**ExtMem 库是为本课程专门开发的模拟外存磁盘块存储和存取的程序库，不保证其能够真正实现对磁盘块的存取操作，同时也不保证其排除一切软件错误。本课程及 ExtMem 开发者不会对使用该程序库所导致的一切错误负责。

## 3.4 数据准备

本实验使用 ExtMem 程序库已预先建立了关系 R 和 S 的物理存储。关系的物理存储形式为磁盘块序列 $B_1, B_2, \cdots, B_n$，其中 $B_i$ 的最后 4 个字节存放 $B_{i+1}$ 的地址。extmem-c\data 下的每个文件模拟一个磁盘块。R 和 S 的每个元组的大小均为 8 个字节。每个磁盘块大小为 64 个字节，可存放 7 个元组和 1 个后继磁盘块地址。

初始化缓冲区块的大小为 64 个字节，缓冲区大小设置为 64*8+8=520 个字节。其中 8 个字节为标志位，表示每个块是否被占用。这样，每块可存放 7 个元组和 1 个后继磁盘块地址，缓冲区内可最多存放 8 个块。

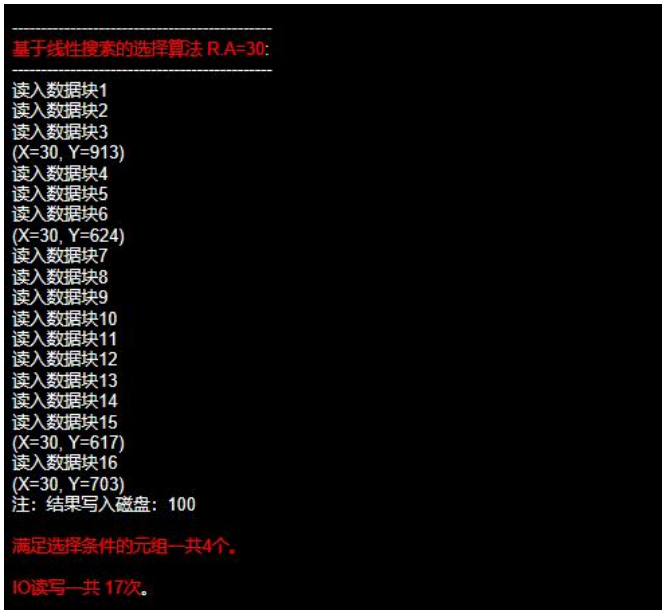
本实验已随机生成关系 R 和 S，R 中包含 16 * 7 = 112 个元组，S 中包含 32 * 7 = 224 个元组。extmem-c\data 目录下文件 1.blk 至 16.blk 为关系 R 的元组数据，文件 17.blk 至 48.blk 为关系 S 的元组数据。

# 4. 范例说明

仅参考输出样式，实验数据已更新！

## 4.1 基于线性搜索的关系选择算法

输出示例：




## 4.2 两阶段多路归并排序算法（TPMMS）

输出示例：
关系 R 排序后输出到文件 301.blk 到 316.blk。
关系 S 排序后输出到文件 317.blk 到 348.blk。

**算法基本思想：**
（1）划分子集并子集排序：将 B 块数据划分成 N 个子集合，使每个子集合块数小于内存缓冲区可用块数。每个子集合装入缓冲区采用内排序排好序并重新写回磁盘。
（2）各子集间归并排序：N 个已排序子集合的数据利用内存缓冲区进行总排序。

已知：$S_{problem}$为待排序元素集合，$R_{problem}$–待排序集合中的元素个数，$R_{block}$-磁盘块或内存块能存储的元素个数，$B_{memory}$-可用内存块的个数， R(S)为求集合 S 的元素个数的函数，$M_i$为内存的第 i 块，$P_{output}$为输出块内存中当前元素的指针。

1. 将待排序集合 $S_{problem}$划分为 m 个子集合 $S_1$, $S_2$,...,$S_m$, 其中 $S_{problem}=\cup_{i=1,...,m}S_i$，且 $R_{problem}=\Sigma_{i=1,...,m}R(S_i)$, $R(S_i)<=B_{memory}*R_{block}$, i=1,...,m(注:每个 $S_i$的元素个数小于内存所能装载的元素个数).
2. **for** i=1 **to** m
3. {将 $S_i$装入内存，并采用一种内排序算法进行排序，排序后再存回相应的外存中}

/*步骤 2 和 3 完成子集合的排序。接下来要进行归并，$M_1$,...,$M_m$用于分别装载 $S_1$,...,$S_m$的一块*/;

4. **for** i=1 **to** m
5. { 调用 read block 函数，读 $S_i$的第一块存入 $M_i$中，同时将其第一个元素存入 $M_{compare}$的第 $i_{th}$个位置; }
6. 设置 $P_{output}$为输出内存块的起始位置;
7. 求 $M_{compare}$中 m 个元素的最小值及其位置 i。
8. **If** (找到最小值及其位置 i) **then**
9. {　　将第 $i_{th}$个位置的元素存入 $M_{output}$中的 $P_{output}$位置，$P_{output}$指针按次序指向下一位置；
10. 　　**If** ($P_{output}$指向结束位置) **then**
11. 　　{　　调用 Write Block 按次序将 $M_{output}$写回磁盘；置 $P_{output}$为输出内存块的起始位置；继续进行; }
12. 　　获取 $M_i$的下一个元素.
13. 　　**If** ($M_i$有下一个元素)
14. 　　{ 将 $M_i$下一个元素存入 $M_{compare}$的第 $i_{th}$个位置。转步骤 7 继续执行。}
15. 　　**Else** { 调用 read block 按次序读 $S_i$的下一块并存入 $M_i$;
16. 　　**If** ($S_i$有下一块)
　　{ 将其第一个元素存入 $M_{compare}$的第 $i_{th}$个位置。转步骤 7 继续执行。}
17. 　　**ELSE** { 返回一个特殊值如 Finished，以示 $S_i$子集合处理完毕，$M_i$为空，且使 $M_{compare}$中的第 $i_{th}$位置为该特殊值，表明该元素不参与 $M_{compare}$的比较操作。转步骤 7 继续执行。}
18. 　　}
19. } /*若 $M_{compare}$的所有元素都是特殊值 Finished，即没有最小值，则算法结束*/

## 4.3 基于索引的关系选择算法

输出示例：


## 4.4 基于排序的连接操作算法（Sort-Merge-Join）

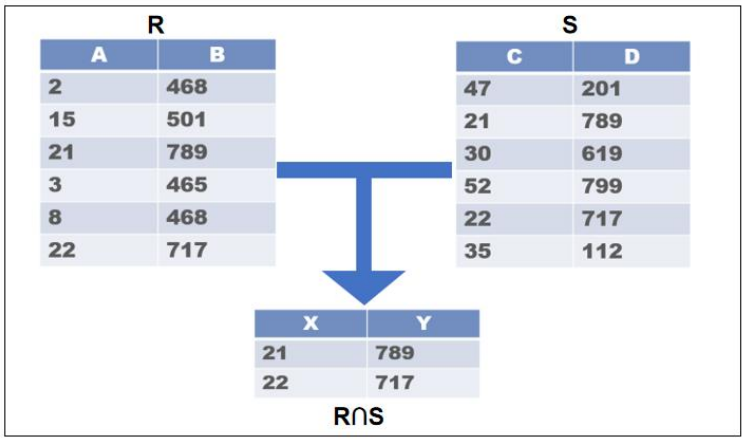
**算法基本思想：**

第一趟：划分 R 和 S 的子表并进行子表排序，排序均基于连接属性排序。
第二趟：归并时注意是 R 的输入还是 S 的输入。R 和 S 的两路输入之间进行连接检查并连接后输出。

输出示例：


## 4.5 集合并、交、差算法

假设 R 有 A、B 两个属性，S 有 C、D 两个属性，R 和 S 满足并相容性。那么 R∩S 的结果如下图：




➢ **基于排序的集合操作算法**

**算法基本思想：**
第一趟：划分 R 和 S 的子表并进行子表排序。
第二趟：归并时注意是 R 的输入还是 S 的输入。R 和 S 的两路输入之间按要求进行输出。

- **基于散列的集合操作算法**

**算法基本思想：**

以连接属性做散列关键字，设计散列函数。

第一趟：将关系 R 和 S 分别使用相同的散列函数 $h_p$ 散列成 M-1 个子表，形成 $R_1,...,R_{M-1}$ 和 $S_1,...,S_{M-1}$，并进行存储。

第二趟：处理每一个子表，用另一个散列函数 $h_r$ 将 $R_i$ 再整体散列读入到内存缓冲区，再依次处理 $S_i$ 的每一块，进行不同操作的处理。

输出示例：

```
----------------------------------------
基于散列的集合的交算法:
----------------------------------------
(X=36,Y=895)
(X=22,Y=712)
(X=30,Y=624)
(X=23,Y=758)
(X=30,Y=703)
(X=30,Y=617)
(X=25,Y=440)
注：结果写入磁盘：140
(X=40,Y=557)
(X=34,Y=665)
注：结果写入磁盘：141

S和R的交集有9个元组。
```

```
----------------------------------------
基于散列的集合的并算法:
----------------------------------------
注：结果写入磁盘：801
注：结果写入磁盘：802
注：结果写入磁盘：803
...
注：结果写入磁盘：845
注：结果写入磁盘：846
注：结果写入磁盘：847

R和S的并集有327个元组。
```

```
----------------------------------------
基于散列的集合的差算法:
----------------------------------------
注：结果写入磁盘：901
注：结果写入磁盘：902
注：结果写入磁盘：903
...
注：结果写入磁盘：928
注：结果写入磁盘：929
注：结果写入磁盘：930
注：结果写入磁盘：931

S和R的差集(S-R)有215个元组。
```

# 5. 参考资料

Abraham Silberschatz, Henry F.Korth. 《数据库系统概念（第六版）》