

实验四 查询处理算法的模拟实现

2021春





本学期实验总体安排

实验项目	一	二	三	四
学时数	2	2	2	2
实验内容	MySQL关系数据库管理系统及SQL语言的使用	一个小型系统的设计与实现 (数据库设计)	一个小型系统的设计与实现	查询处理算法的模拟实现
分数	4	4	6	6
检查方式	课堂抽查、提交实验截图	提交模型图	提交实验报告、工程文件、系统介绍录频	提交实验报告、工程文件

本学期实验课程共 **8** 个学时，**4** 个实验项目，总成绩为 **20** 分。



目录

1

实验目的

2

预备知识

3

实验内容

4

思考题

1

实验任务

2

实验数据

3

实验要求

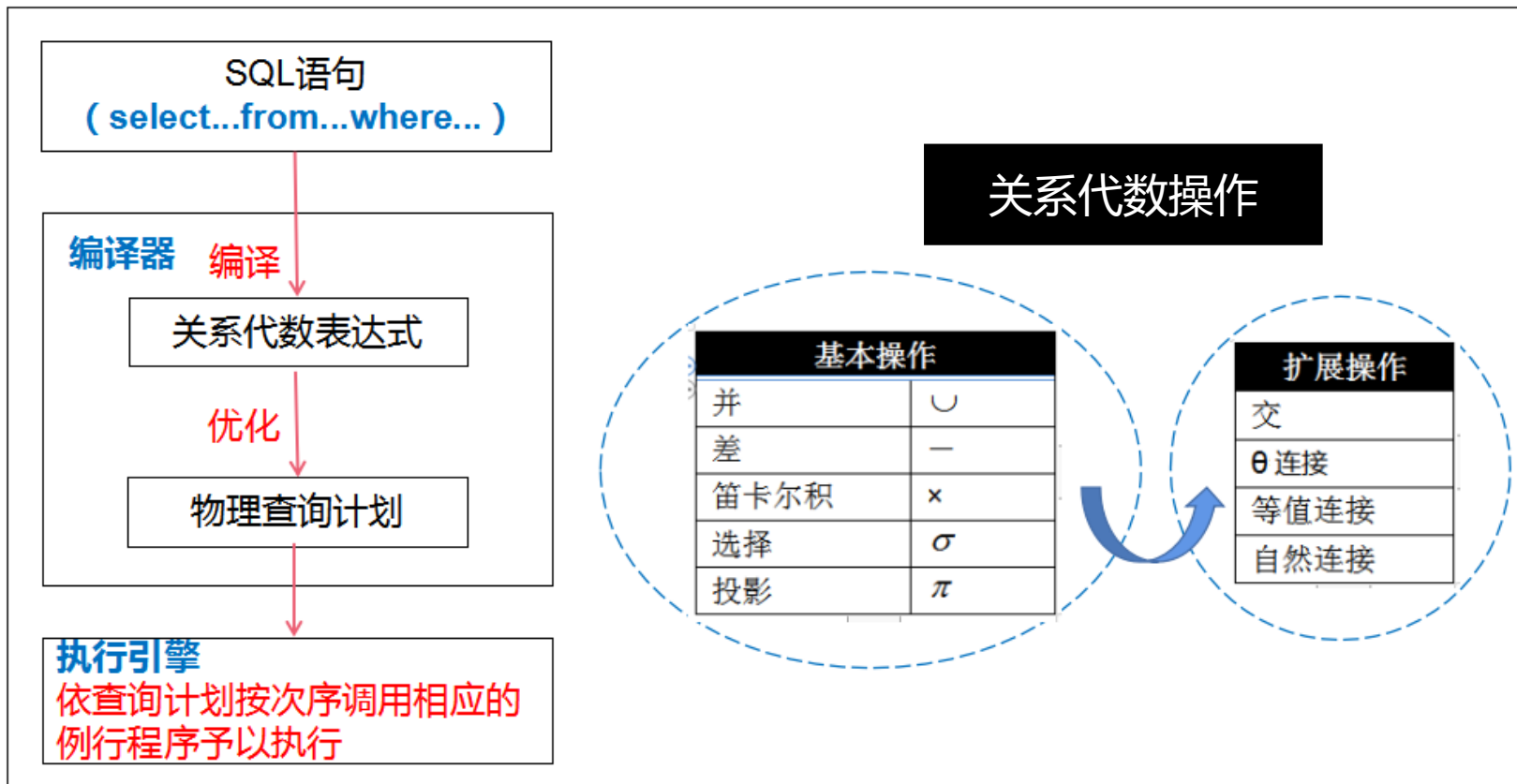


实验目的

- 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；
- 加深对算法I/O复杂性的理解。

预备知识

DBMS查询实现的基本思想

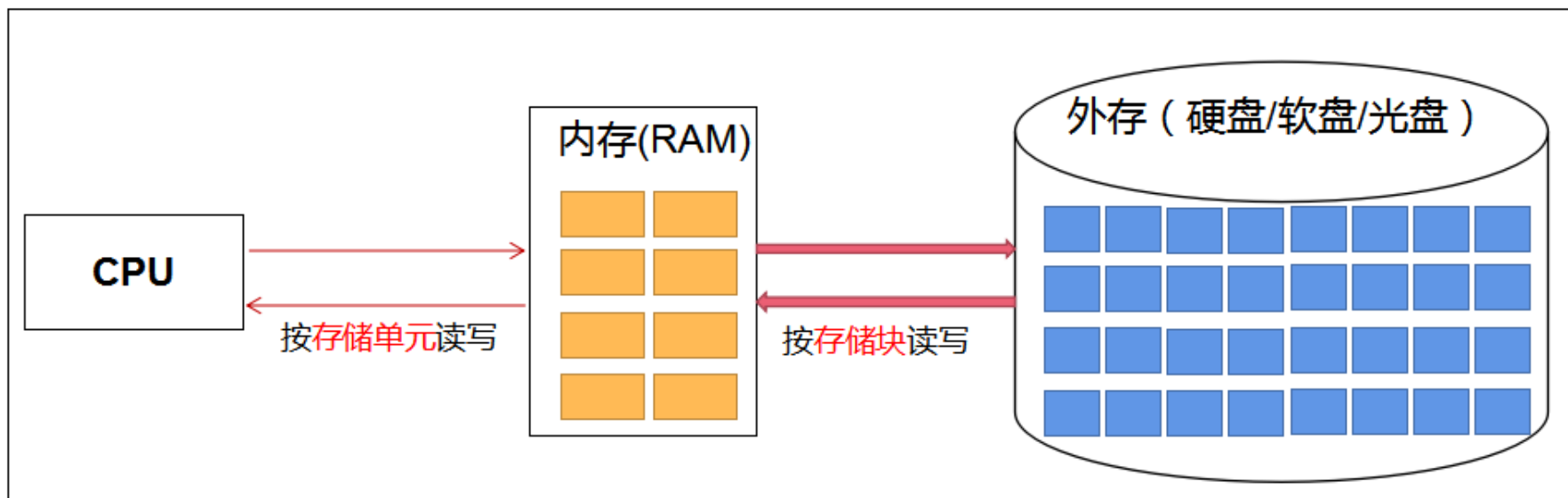




预备知识

CPU、内存和外存的关系

- CPU与内存直接交换信息，按存储单元（存储字）进行访问
- 外存按存储块进行访问，其信息需现装入内存，才能被CPU处理





预备知识

索引

定义在存储表基础上，有助于无需检查所有记录而快速定位所需记录的一种辅助存储结构。





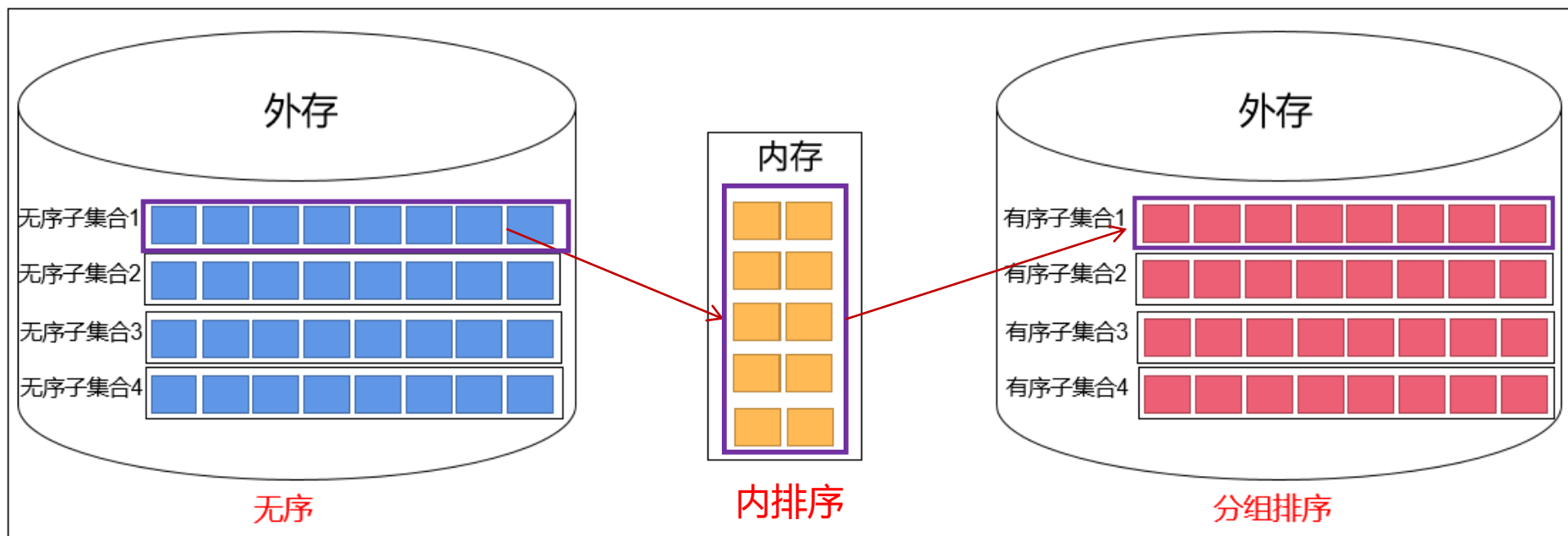
预备知识

两阶段多路归并排序算法 (TPMMS)

待排序数据不能一次装入内存，需将数据分批装入分批处理。

基本思想：

(一) 划分子集并子集排序

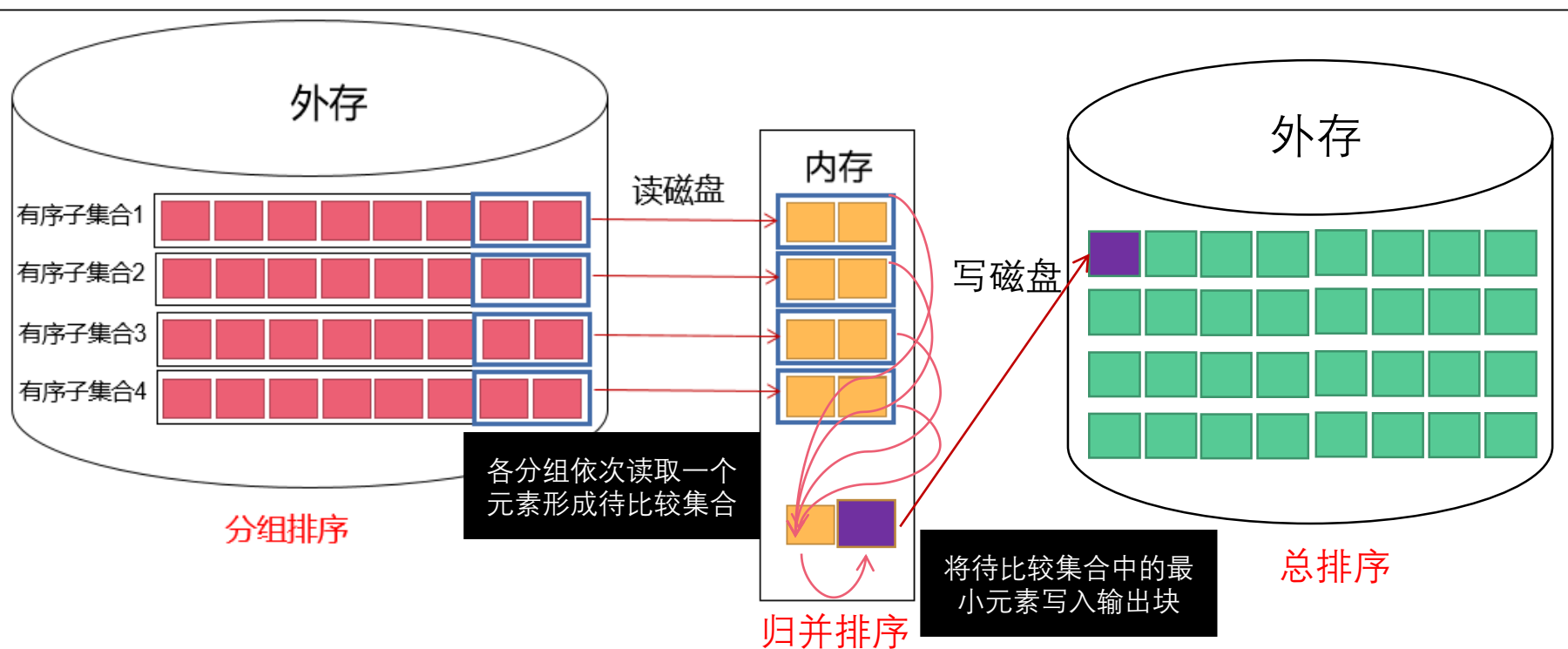




预备知识

两阶段多路归并排序算法 (TPMMS)

(二) 各子集间归并排序



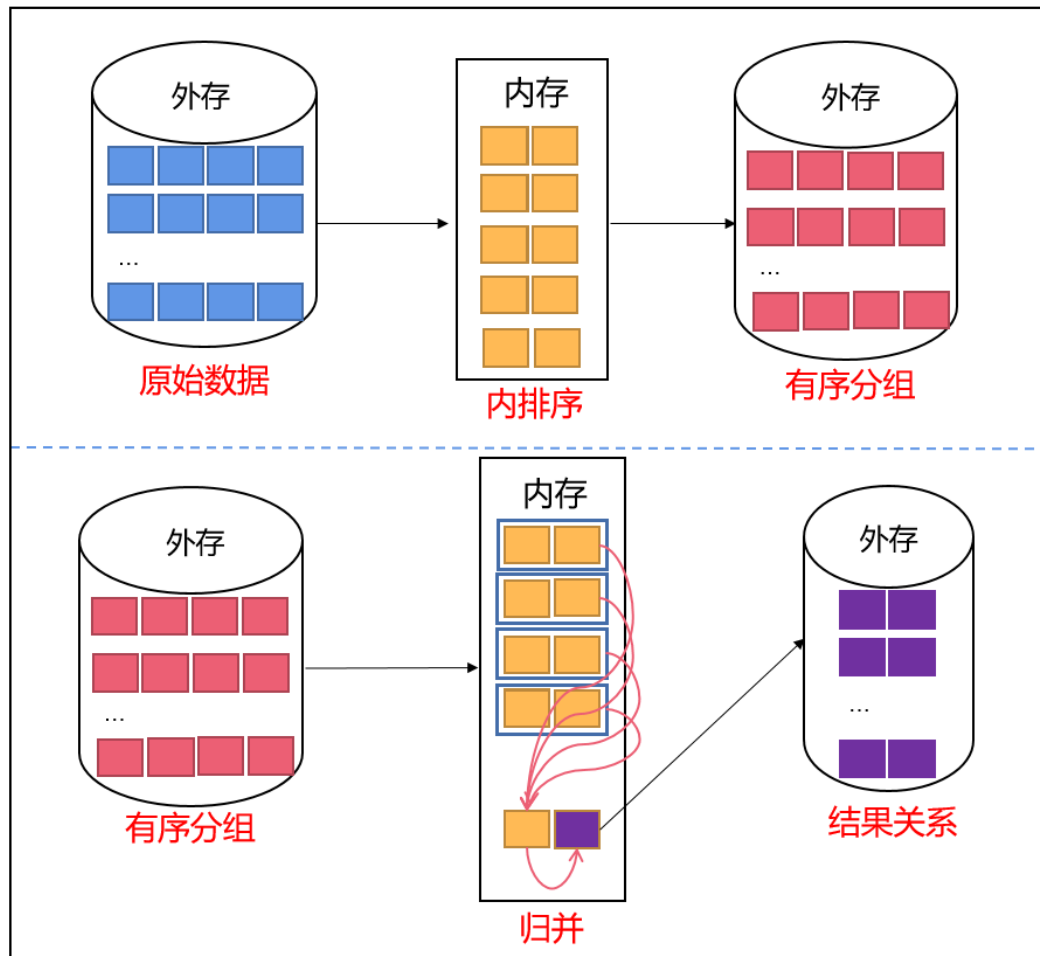
预备知识

基于排序的两趟扫描算法

第一趟：划分子表并进行子表排序

第二趟：归并阶段

- 关系一元操作
 - 去重复
 - 排序
 - 分组聚集
- 关系二元操作
 - 连接
 - 集合的并、交、差



预备知识

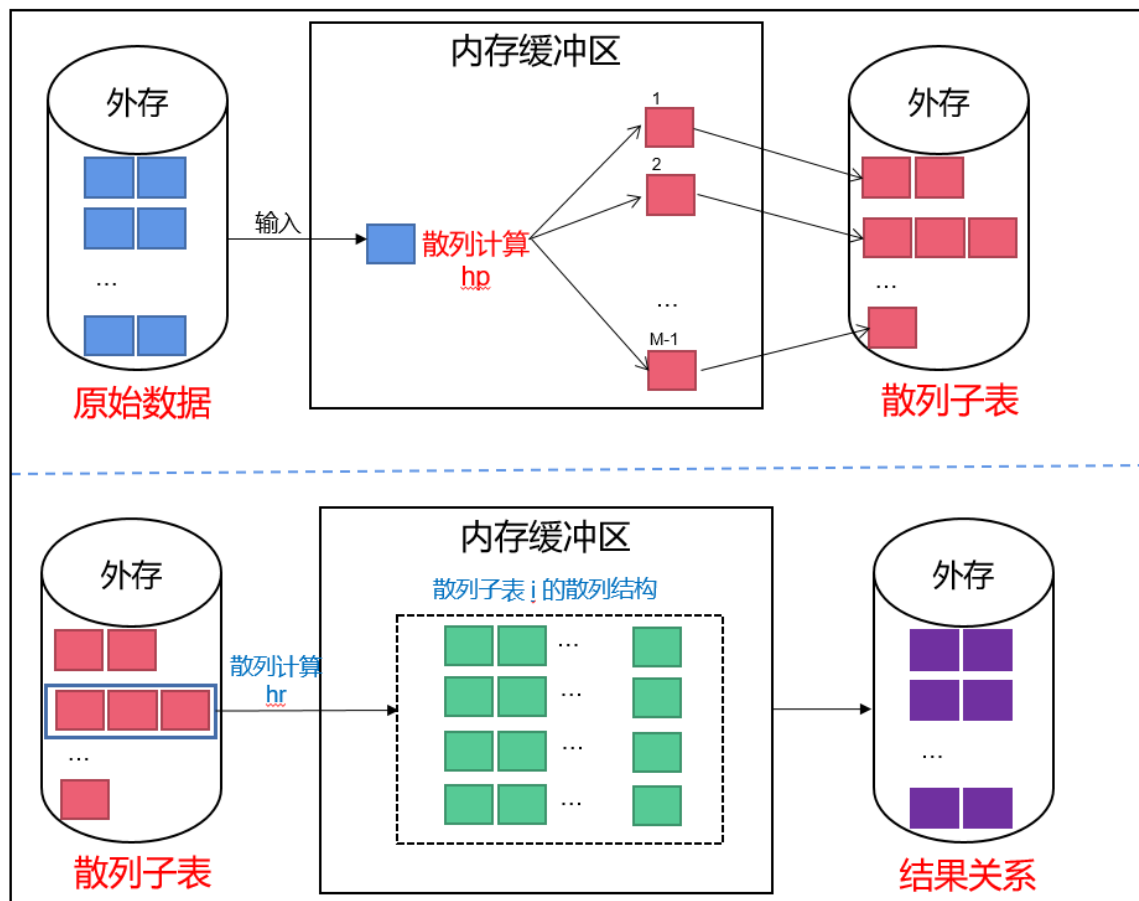
基于散列的两趟扫描算法

第一趟：散列子表

- 用散列函数 hp 将原始关系划分到 $M-1$ 个子表并存储， M 为内存块数。

第二趟：处理每个子表

- 用另一个散列函数 hr 将子表读入内存，进行不同操作的处理。





实验任务

1. 基于ExtMem程序库，实现关系选择、连接等操作算法
2. 实现集合并、交、差操作算法

实验要求：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。



实验数据

- ① **关系R**具有两个属性A和B， A的值域为[20, 60]， B的值域为[1000, 2000];
- ② **关系S**具有两个属性C和D， C的值域为[40, 80]， D的值域为[1000, 3000]。

例：R的元组 (30, 1027) 表示 $R.A = 30$; $R.B = 1027$ 。

A	B
30	1027
45	1089
...	...

注：属性值均为int型（4个字节）， R和S的每个元组的大小均为8个字节。

3	0	\0	\0	1	0	2	7
---	---	----	----	---	---	---	---



实验数据

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下的每个文件模拟一个磁盘块。

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。

(21, 1320)	2	1	\0	\0	1	3	2	0
(22, 1034)	2	2	\0	\0	1	0	3	4
(25, 1111)	2	5	\0	\0	1	1	1	1
(34, 1309)	3	4	\0	\0	1	3	0	9
(36, 1547)	3	6	\0	\0	1	5	4	7
(36, 1512)	3	6	\0	\0	1	5	1	2
(55, 1345)	5	5	\0	\0	1	3	4	5

↓
2.blk

关系R:

1.blk	2.blk	3.blk	4.blk	5.blk	6.blk	7.blk	8.blk
9.blk	10.blk	11.blk	12.blk	13.blk	14.blk	15.blk	16.blk

关系S:

17.blk							
							48.blk

文件夹extmem-c\data (模拟磁盘)



ExtMem程序库

ExtMem是C语言开发的模拟外存磁盘块存储和存取的程序库。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。

1个数据结构:

Buffer数据类型:

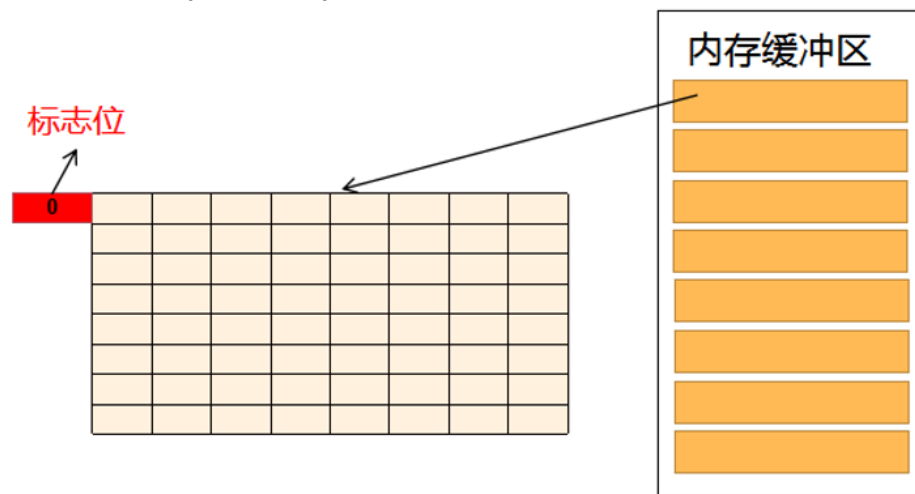
numIO: 外存I/O次数
bufSize: 缓冲区大小 (单位: 字节)
blkSize: 块的大小 (单位: 字节)
numAllBlk: 缓冲区内可存放的最多块数
numFreeBlk: 缓冲区内可用的块数
data: 缓冲区内内存区域(char *)

缓冲区最多存放**8个块**: $64 * 8 + 8 = 520$
其中8个字节为**标志位**, 表示每个块是否被占用。

内存缓冲区

缓冲区大小 (bufSize) : 520个字节

块的大小 (blkSize) : 64个字节





ExtMem程序库

文件test.c中给出了API函数使用的具体示例:

7个API函数:

函数	功能
initBuffer	初始化缓冲区buf
getNewBlockInBuffer	在缓冲区中申请一个新的块
readBlockFromDisk	将磁盘上地址为addr的磁盘块读入缓冲区buf
writeBlockToDisk	将缓冲区buf内的块blk写入磁盘上地址为addr的磁盘块
freeBlockInBuffer	解除块对缓冲区内存的占用
dropBlockOnDisk	从磁盘上删除地址为addr的磁盘块内的数据
freeBuffer	释放缓冲区buf占用的内存空间

```
int main(int argc, char **argv)
{
    Buffer buf; /* A buffer */
    unsigned char *blk; /* A pointer to a block */
    int i = 0;

    /* Initialize the buffer */
    if (!initBuffer(520, 64, &buf))
    {
        perror("Buffer Initialization Failed!\n");
        return -1;
    }

    /* Get a new block in the buffer */
    blk = getNewBlockInBuffer(&buf);

    /* Fill data into the block */
    for (i = 0; i < 8; i++)
        *(blk + i) = 'a' + i;

    /* Write the block to the hard disk */
    if (writeBlockToDisk(blk, 8888, &buf) != 0)
    {
        perror("Writing Block Failed!\n");
        return -1;
    }

    /* Read the block from the hard disk */
    if ((blk = readBlockFromDisk(1, &buf)) == NULL)
    {
        perror("Reading Block Failed!\n");
        return -1;
    }

    /* Process the data in the block */
    int X = -1;
    int Y = -1;
    int addr = -1;

    char str[5];
    printf("block 1:\n");
    for (i = 0; i < 7; i++) //一个blk存7个元组加一个地址
    {
        for (int k = 0; k < 4; k++)
        {
            str[k] = *(blk + i*8 + k);
        }
        X = atoi(str);
        for (int k = 0; k < 4; k++)
        {
            str[k] = *(blk + i*8 + 4 + k);
        }
        Y = atoi(str);
        printf("%d, %d ", X, Y);
    }
}
```




实验要求

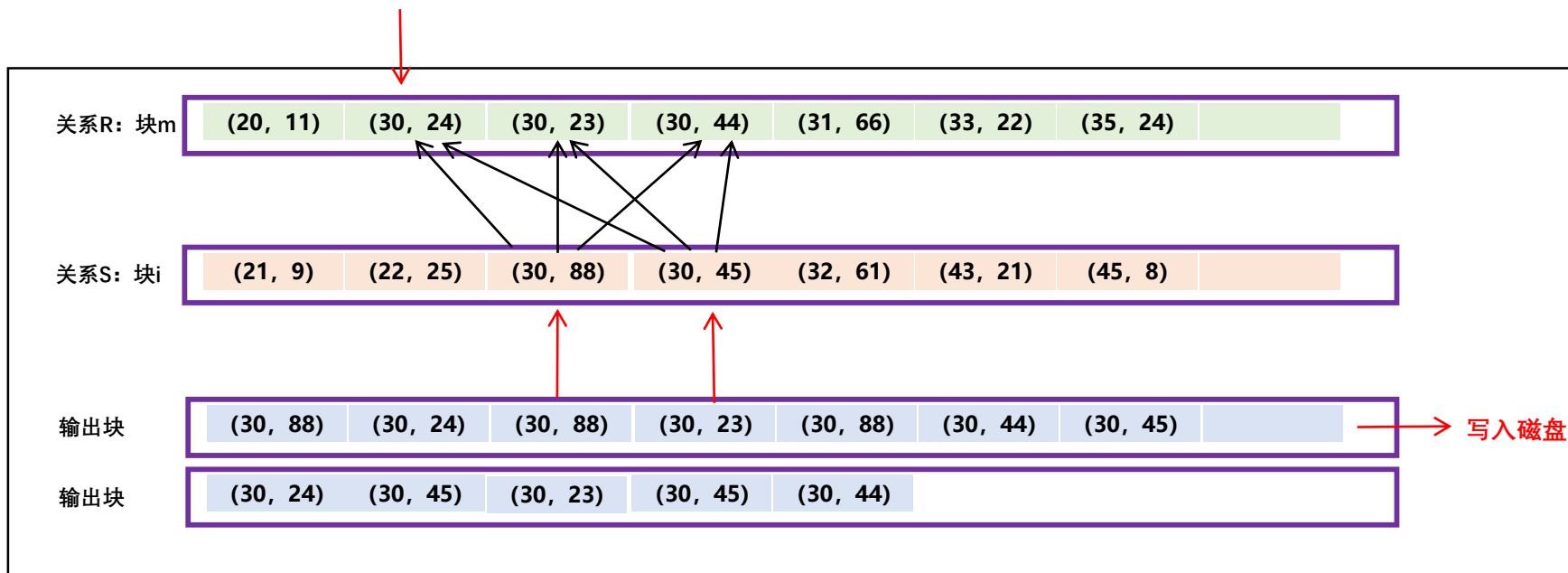
基于ExtMem程序库，使用C语言模拟实现以下关系操作：

- 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 $S.C=50$ 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 = 50`）
- 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。（不可定义长度大于10的整型或字符型数组）
- 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 $S.C=50$ 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 = 50`）
- 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（ $S \cup R$ ）、交（ $S \cap R$ ）、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。



实验要求

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





输出范例

基于线性搜索的选择算法 R.A=30:

读入数据块1
读入数据块2
读入数据块3
(X=30, Y=913)
读入数据块4
读入数据块5
读入数据块6
(X=30, Y=624)
读入数据块7
读入数据块8
读入数据块9
读入数据块10
读入数据块11
读入数据块12
读入数据块13
读入数据块14
读入数据块15
(X=30, Y=617)
读入数据块16
(X=30, Y=703)
注: 结果写入磁盘: 100

满足选择条件的元组一共4个。

IO读写一共 17次。

基于索引的选择算法 R.A=30:

读入索引块350
没有满足条件的元组。
读入索引块351
读入数据块312
(X=30, Y=913)
(X=30, Y=624)
(X=30, Y=617)
读入数据块313
(X=30, Y=703)
注: 结果写入磁盘: 120

满足选择条件的元组一共4个。

IO读写一共 5次。

对比IO读写次数，理解索引的作用。

仅参考输出样式，
实验数据已更新！



输出范例

基于排序的连接算法：

注：结果写入磁盘：701
注：结果写入磁盘：702
...
注：结果写入磁盘：760
注：结果写入磁盘：761
注：结果写入磁盘：762
注：结果写入磁盘：763

总共连接220次。

基于排序的集合的交运算：

(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。

基于排序的集合的并运算：

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。

基于排序的集合的差运算：

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。

仅参考输出样式，
实验数据已更新！



思考题

- ★ 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：
并、交、差 (S-R) 。（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。



评分标准

基本任务 (100 分)

- ① 基于线性搜索的关系选择 (10分)
- ② 两阶段多路归并排序算法 (40分)
- ③ 基于索引的关系选择算法 (20分)
- ④ 基于排序的连接操作算法 (20分)
- ⑤ 并、交、差其中一种集合操作算法 (10分)

附加题 (10 分)

并、交、差剩余两种集合操作算法 (每种算法5分, 选做)



提交方式

➤ **课后提交：**将实验报告、工程文件打成 rar 或者 zip 包，提交到平台：<http://lab.hitsz.edu.cn/>

注意：

1. 强烈建议使用firefox、chrome浏览器；
2. 用户名、密码为统一身份认证；
3. 请提交到相应的条目；
4. **统一命名：**学号_姓名_数据库实验四
5. 实验报告需有核心代码讲解、运行结果截图、遇到并解决的问题等。
6. 提交**截止时间：**实验课后两周（具体时间看平台）



**同学们
请开始实验吧!**