数据库管理系统原理与实现 实验报告

实验一 存储管理

组长：金凤美
组员：岳永姣、孙殿森、徐茂增、王皓

一、 实验内容

- ✧ 数据文件的组织
- ✧ 缓冲区管理
- ✧ 空闲空间管理

二、 实验要求

1. 数据文件的组织
   a) 段页式组织方式
   b) 支持基本数据类型，可不支持大对象数据
2. 缓冲区管理
   a) 缓冲区页面组织
   b) 缓冲区查找
   c) 缓冲区淘汰
3. 空闲空间管理
   a) 空闲空间组织
   b) 空闲空间分配
   c) 空闲空间回收

三、 实验环境

1. 操作系统平台：macOS Sierra
2. 编程语言：C++
3. 程序开发环境：Xcode 7.0

四、 实验原理

1. 数据库管理系统

数据库管理系统(Database Management System)是一种操纵和管理数据库的大型软件，用于建立、使用和维护数据库，对数据库进行统一的管理和控制，以保证数据库的安全性和完整性。用户通过DBMS访问数据库中的数据，数据库管理员也通过DBMS进行数据库的维护工作。它可使多个应用程序和用户用不同的方法在同时或不同时刻去建立、修改和访问数据库。大部分DBMS提供数据定义语言DDL（Data Definition Language）和数据操作语言DML（Data Manipulation Language），供用户定义数据库的模式结构与权限约束，实现对数据的追加、删除等操作。

数据库管理系统是数据库系统的核心，是管理数据库的软件。数据库管理系统就是实现把用户意义下抽象的逻辑数据处理，转换成为计算机中具体的物理数

据处理的软件。有了数据库管理系统，用户就可以在抽象意义下处理数据，而不必顾及这些数据在计算机中的布局和物理位置。

2. DBMS 的层次结构

根据处理对象的不同，DBMS的层次结构由高级到低级依次为应用层、语言翻译处理层、数据存取层、数据存储层、操作系统。

- ✧ 应用层：DBMS与终端用户和应用程序的界面层，处理的对象是各种各样的数据库应用。
- ✧ 语言翻译处理层：对数据库语言的各类语句进行语法分析、视图转换、授权检查、完整性检查等。
- ✧ 数据存取层：将上层的集合操作转为单记录操作。处理对象是单个元组。
- ✧ 数据存储层：处理的对象是数据页和系统缓冲区，即本次实验一的重点。
- ✧ 操作系统：DBMS的基础，提供的存取原语和基本的存取方法通常是作为和DBMS存储层的接口。

3. 数据组织、存储于管理

在数据存储层上，DBMS 要分类组织、存储和管理各种数据，包括数据字典、用户数据、存取路径等，需确定以何种文件结构和存取方式在存储级上组织这些数据、如何实现数据之间的联系。数据组织和存储的基本目标是提高存储空间利用率，选择合适的存取方法提高存取效率。

4. 缓冲区管理

密集的磁盘 I/O 操作是数据库引擎的一大特点。而完成磁盘 I/O 操作要消耗许多资源，且耗时较长。缓冲区管理是实现高效 I/O 操作的关键环节。缓冲区管理器负责将数据页或索引页从数据库磁盘文件读入缓冲区高速缓存中，并将修改后的页写回磁盘。缓冲区缓存中会保留一页，直到缓冲区管理器需要该缓冲区读入更多的数据。数据只有在被修改后才会重新写入磁盘。在将缓冲区高速缓存中的数据写回磁盘之前，可对其进行多次修改。

5. 空闲空间管理

空闲空间管理主要是为了实现存储空间的分配和回收，DBMS 会为存储空间设置相应的结构以记住存储空间的使用情况，并配以相应算法方便地对存储空间进行分配和回收。常见的空闲空间管理方法主要有空闲表法、位示图法、空闲块链表法、链接索引块法。

五、 程序设计

1. 数据组织方式

在我们所实现的 DBMS 中（以下简称 ourDBMS），采用的是“记录-页-文件”这样的组织结构层次来对数据进行组织、存储等操作。

【文件】

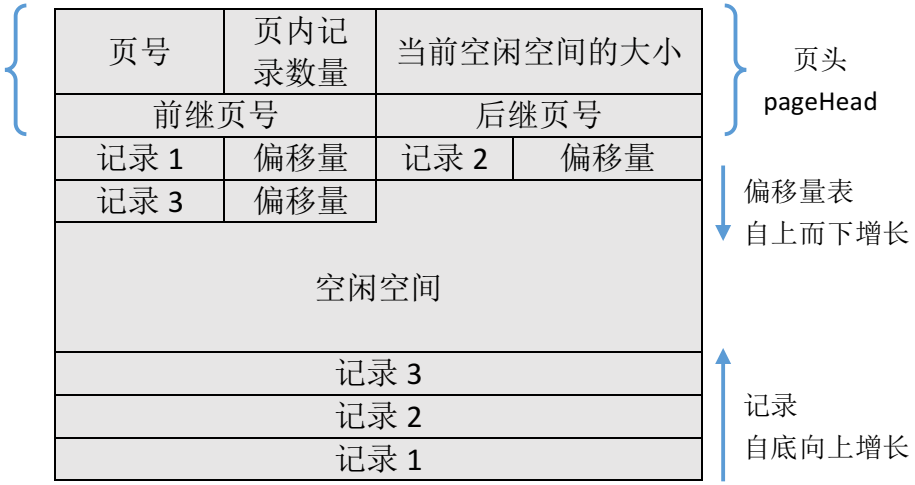
在 ourDBMS 中，为当前数据库用户所创建的每个表都建立一个文件，每个文件的相关信息会记录在系统头部(struct FileDesc)，同时也会记录将当前的文件数量等参数记录在数据库头部(struct dbSysHead)中。

初始建立文件时，会分配一个页给文件。当向某一文件（表）中插入大量数据而导致空间不够用时，DBMS 会动态地为其扩展空间，即再给这个文件分配连续的空闲页，并将新页链接到当前文件所占有的页的最尾部，因此一个文件所占用的页可能不连续。

当删除一个文件时，需要回收它的文件号，以及它所占用的每个页，即在空闲空间位示图中重置标记位。

【页/块】

页/块是 I/O 操作的最小单位，页和块是不同情况下的表述，实际大小都规定为 4KB，在磁盘中称为页，在内存、缓冲区中则称为块。每一页的组织方式如下所示，共包含三大部分：页头、偏移量表和记录。



其中，每页有一个固定大小的页头(struct pageHead)，记录了该页的页号、页内包含的记录数量、当前的空闲空间大小，以及前继页号和后继页号。这样的定义是因为文件的大小（即所占用的页数）是动态增长的，因而在文件的相关信息中，ourDBMS 只需记录该文件的起始页号，然后通过读页头信息中的后继页号来找到下一页，以此类推，用这种顺序的方式来找到属于该文件的所有页。

从页尾开始，逐条插入记录，同时会为它分配一条偏移量表项，存储了它在当前页内的记录号和相对于页尾地址的偏移量大小，并将该偏移量表项从页头结尾处开始，依次向后面的空闲空间中插入。这样，通过偏移量表就可以读取页内的每条记录了。宏观地来看，只要给定页号和页内记录号，ourDBMS 就可以唯一地确定一个物理地址，进而找到一条记录。

【记录】

因为用户在数据库中创建表时会自行规定每条记录的模式，即每个字段的类型、大小等，计划是令 ourDBMS 将当前数据库中创建的所有表模式单独存储到一个文件中。用户在对数据进行操作时，例如执行 select name from table1 的语句，首先会去找到 table1 对应的表模式，然后进行字段解析等相关操作，得到用户指定的字段，再去对数据进行读取等。

目前的进度是在进行读写文件时，会将所有数据预处理成字符串形式，再作为参数传入，然后 ourDBMS 会进行相关读写操作。但上述功能是一个数据库管理系统必须要实现的，因此也是我们接下来的重点。

2. 映射表（待实现）

映射表的存在是为了方便进行 b+树索引的相关操作，尤其是在修改索引条目时。在建立索引的过程中，每个叶节点存储的是一个 key 值和对应在映射表中的逻辑地址。这样，如果给定 key 值去查找对应的记录，首先通过索引取出逻辑地址，再去映射表里找其对应的物理地址。

当对表中的数据进行了增删改查等操作时，可能会出现很多条记录的物理地址发生改变。如果没有映射表，系统就需要在索引文件中找到所有发生改变的记录，将其改为新的物理地址。但有了映射表就简化了该步骤，只需要单独修改映射表中相关记录的项即可。

如下图所示，目前的设想是对每个存储到该数据库中的记录赋一个逻辑号，即为 DB 逻辑地址，这个逻辑号是不会重复的。而由上述可知，给定一个页号和页内记录号也可以唯一地标识一条记录，这样“逻辑号”和“页号+页内记录号”就构成了一种一一对应的关系，类似操作系统中的“逻辑地址”和“物理地址”。当移动了一个文件中的某条记录时，不论是页内移动还是将其放到了该文件下的其他页中，只需要修改表内的物理地址即可，其他任何指向该记录的“指针”（索引文件等）都不需要被修改。

为了减少 I/O 操作、提高读写效率，一个数据库的映射表是长时间放在内存中的，只有在关闭该数据库时，才会将映射表重新写回数据库中，ourDBMS 为映射表单独创建一个文件，与其他用户表文件的管理、操作方式相同。

另外，映射表中还有一个标记位，用于标识当前记录是否被删除。这是为了便于逻辑号的分配与回收。当要删除某条记录时，暂时不需要删除其在映射表中的表项，而是将此标记位置为 1，表示这条记录已经被删除了。当一个指针通过逻辑号在映射表中找到了这条记录，想读取其物理地址时，系统会先查看此标记位，若该条记录为 1，则拒绝读取，因为这条记录已经被删除了，其原物理地址有可能已经存储了其他数据。

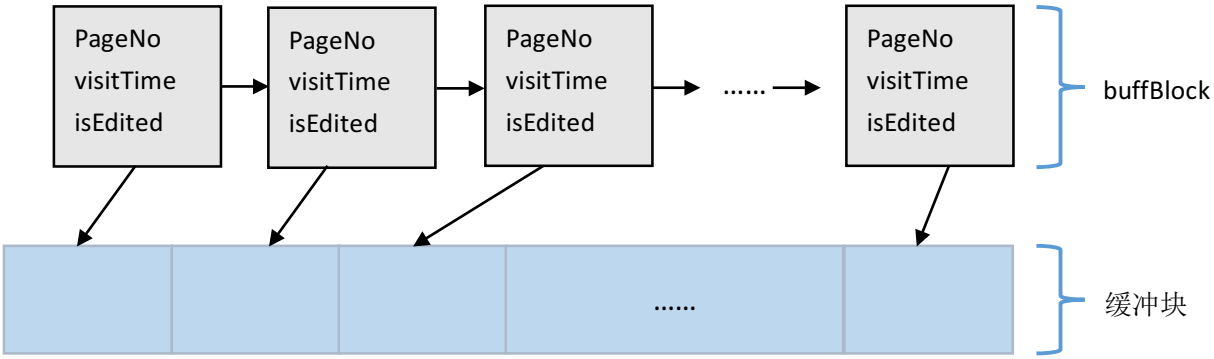
逻辑号的回收方式还没有想好，目前想法是建立一个类似于空闲页的位示图，但由于逻辑号的数量并不确定，而是动态增长的，因而觉得也不是最佳办法。另外，对于映射表的整体设计也仍存疑，不知道理解是否有误，以及这样定义和构建是否正确。

映射表			
是否删除	DB 逻辑地址（逻辑号）	DB 物理地址	
		页号	页内记录号
1	1	0	0
0	2	0	1
1	3	0	2
1	4	1	0
0	5	1	1
0	6	1	2
0	7	2	0
…	…	…	…

3. 缓冲区管理

ourDBMS 为每个数据库分配一定大小的内存作为缓冲区，内存的管理单元大小与磁盘存取粒度保持一致，即缓冲区块与页大小相同。另外，为了管

理缓冲区，需要维护一个数组，每个元素是一个描述块(struct buffBlock)，与一个缓冲区块相对应，每个描述块记录着其对应的缓冲区块中存放的页号、这一页停留在内存中的时间以及其是否被编辑过。



4. 定义结构体

```
//数据库系统的相关参数、空闲空间位示图、缓冲区、文件指针
struct dbSysHead{
    struct SysDesc desc;
    unsigned long *FreeSpace_bitMap;
    struct buffSpace buff;
    FILE *fpdesc;
};
```

```
// 数据库系统的一些参数（有一些是固定值，但有一些会改变）
struct SysDesc{
    long sizeOfFile;            //  文件中数据区的大小，默认设为196MB
    long sizePerPage;           //  每一个页的大小，默认4KB
    long totalPage;             //  总共的页数
    long pageAvai;              //  当前有多少可用的页
    //bitMap 空闲空间位示图
    long bitMapAddr;                //  文件中bitMap的起始地址
    long sizeBitMap;                //  bitMap的大小，以字节为单位
    long dataAddr;                  //  文件中数据区的起始位置
    int curFileNum;         //目前有多少个文件，最多为 MAX_FILE_NUM
    struct FileDesc fileDesc[MAX_FILE_NUM]; //对每一个文件进行描述
};
```

```
//描述文件的相关信息
struct FileDesc{
    int fileType;//标识是否为索引文件，以及索引文件的类型（hash/b树/顺序）
    int fileID;   //文件号
    long fileFirstPageNo;    //该文件所占用的第一个页的页号
    long filePageNum;        //文件占用了多少页
};
```

```
//页头信息，放在每页的起始位置
struct pageHead{
    long pageNo;                    // 页号
    int curRecordNum;               // 当前该页存储的记录个数
    long prePageNo;                 // 该页所属的文件
    long nextPageNo;
    long freeSpace;                 // 该页的空余空间大小
};
//页内偏移量表的每一项
//块内的偏移量表从块的前端向后增长，块内的记录是从后向前放置
struct offsetInPage{
    int recordID;       // 记录在本页内的记录号
    int offset;         // 该记录相对于块尾地址的偏移量
    bool isDeleted;     // 这条记录是否已被删除
};
```

```
//缓冲区空间的定义
struct buffSpace{
    struct buffBlock map[SIZE_BUFF];
    char data[SIZE_BUFF][SIZE_PER_PAGE];
    long curTimeStamp;
};
//记录每个缓冲区块的相关信息
struct buffBlock{
    long pageNo;
    long visitTime;
    bool isEdited;
};
```

尚未使用、待实现具体函数的结构体

```
//数据库映射表，用于存放每条记录的逻辑地址和物理地址，在索引部分有应用
struct dbMap {
    long logicID;
    long pageNo;
    long recordID;
};
```

与 B+树索引的相关结构体，后期需要与实验二结合起来

```
//b+树中每个节点所包含的记录
//在叶节点中表示数据记录，内部节点中表示键值-指针对
typedef struct {
    int key;        //节点中的一个键值
    int pos;        // 指向的子节点的偏移量或键值为key的记录的逻辑地址
```

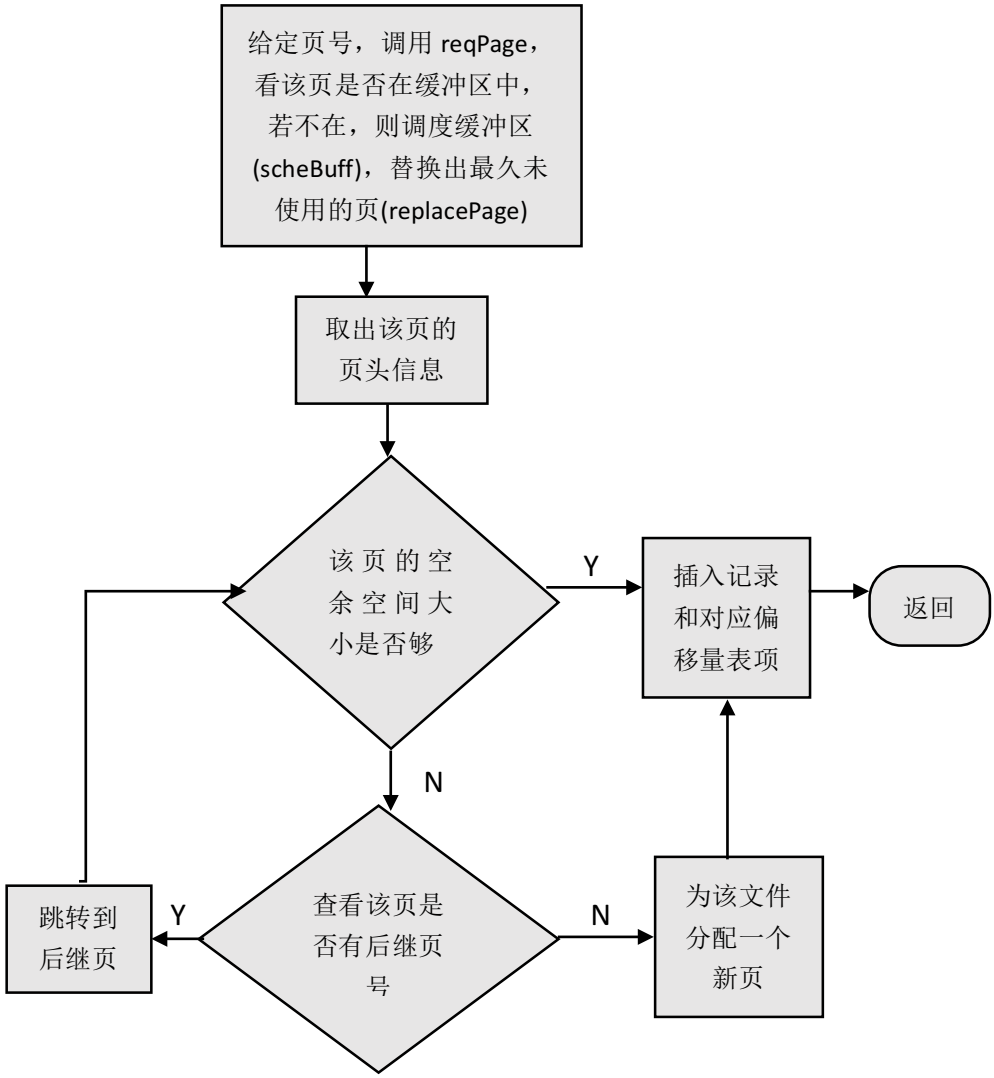
```
}Record;

//b+树的节点定义
typedef struct {
    int type;          // 节点类型，内部节点或叶节点
    int count;         // 节点中的Record个数
    int parent;        //该节点的父节点在索引文件中的偏移量
                       //若为0，则表示它是根节点
    Record pair[MAX];   //存储该节点的所有Record
}Node;
```

5. 相关函数与功能模块

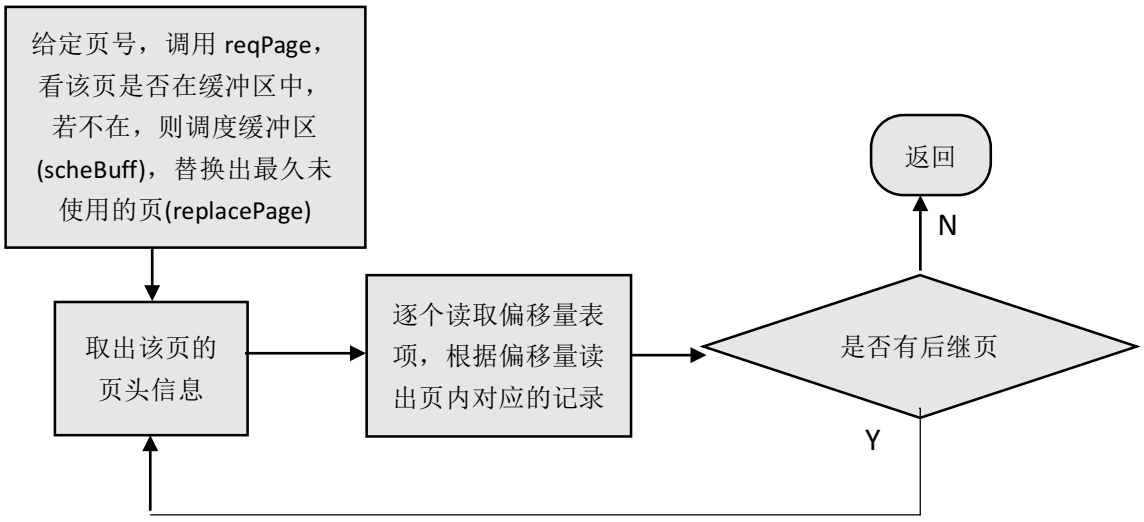
```
//initial.cpp
void createSysHead(char *filename);
void init_database(struct dbSysHead *, char *filename);
void show_SysDesc(struct dbSysHead *);

//fileOption.cpp
int createFile(struct dbSysHead *, int type ,long reqPageNum);
int queryFileIndex(struct dbSysHead *, int fid);
void writeFile(struct dbSysHead *, int fid, int length, char *);
void readFile(struct dbSysHead *, int fid, char *des);
void deleteFile(struct dbSysHead *, int fid);

//buffManage.cpp
int queryPage( struct dbSysHead *head, long queryPageNo );
int replacePage(struct dbSysHead *, int mapNo, long pageNo );
int scheBuff( struct dbSysHead *head );
int reqPage( struct dbSysHead *head, long queryPageNo );

//pageOption.cpp
int getBit(unsigned long num, long pos);
int setBit( unsigned long *num, long pos, int setValue);
long allocatePage( struct dbSysHead *head, long reqPageNum);
void recyOnePage( struct dbSysHead *head ,long pageNo);
void recyAllPage(struct dbSysHead *head);
```

- 数据库初始化(init_database)：从给定的文件中读取数据库的相关参数，若文件不存在，则创建一个数据库头(createSysHead)，并设置相关参数，写回文件。
- 创建文件(createFile)：给定文件类型（可能为用户表文件、索引文件、存储表模式的文件，暂时是只考虑了用户文件），为其分配指定数量的页(allocatePage)，返回的是分配的文件号。在分配页的过程中，重点需要设置页头信息，记录前继、后继页号等，便于后续的文件管理。

- 向文件中写记录(writeFile)：给定文件号，在调用函数之前会对记录进行预处理，然后将其以字符串形式传入，并给定长度，写入该文件的合适位置中。向指定文件中插入一条记录的流程如下：



- 读取文件中的所有记录(readFile)：



- 删除文件(deleteFile)：遍历文件的所有页，将其在空闲空间位示图 bitMap 中的标记位重置，回收文件号，更新数据库系统的头部参数。
- 与单条记录相关的增删改查操作：待实现…

六、 实验结果

1. 初始化数据库，下图为一些参数。

```
**************************************************

sizeOfFile:       205520896
sizePerPage:      4096
totalPage:        50176
pageAvai:         50176
bitMapAddr:       1024
sizeBitMap:       6272
dataAddr:         7296
curFileNum:       0

**************************************************
```

2. 创建文件后，系统参数的变化

```
创建文件0成功!
**************************************************

sizeOfFile:       205520896
sizePerPage:      4096
totalPage:        50176
pageAvai:         50175
bitMapAddr:       1024
sizeBitMap:       6272
dataAddr:         7296
curFileNum:       1

**************************************************
```

3. 插入了 20 万条记录后，文件 0 的一些参数，以及系统的一些参数变化

```
**************************************************

sizeOfFile:       205520896
sizePerPage:      4096
totalPage:        50176
pageAvai:         48620
bitMapAddr:       1024
sizeBitMap:       6272
dataAddr:         7296
curFileNum:       1

**************************************************
```

```
**********************************************************

文件类型:用户表文件
该文件一共占用：1556页
该文件的起始页页号为：0
**********************************************************
```

4. 按页读取文件 0 存储的全部记录
   注：插入的记录为一个结构体，因而在下面的截图中，“记录内容”部分的前几位即为 rid，插入时是从 0 到 199999 按顺序递增的，因此可以作为记录数量的参考。未全部截图，只随机截了中间页 690 和尾页 1555

```
struct employee{
    long rid;
    char name[100];
    int age;
    int wage;
};
```

```
--------页头信息-------
页号：690
页的空余空间：6
该页当前存储的记录个数：135
偏移量表-----记录内容
记录号    偏移量    是否删除
0       18      0   94033abc9406399033
1       36      0   94034abc9406499034
2       54      0   94035abc9406599035
3       72      0   94036abc9406699036
4       90      0   94037abc9406799037
5       108     0   94038abc9406899038
6       126     0   94039abc9406999039
7       144     0   94040abc9407099040
8       162     0   94041abc9407199041
9       180     0   94042abc9407299042
10      198     0   94043abc9407399043
11      216     0   94044abc9407499044
12      234     0   94045abc9407599045
13      252     0   94046abc9407699046
14      270     0   94047abc9407799047
15      288     0   94048abc9407899048
16      306     0   94049abc9407999049
17      324     0   94050abc9408099050
18      342     0   94051abc9408199051
19      360     0   94052abc9408299052
20      378     0   94053abc9408399053
21      396     0   94054abc9408499054
22      414     0   94055abc9408599055
23      432     0   94056abc9408699056
24      450     0   94057abc9408799057
25      468     0   94058abc9408899058
26      486     0   94059abc9408999059
27      504     0   94060abc9409099060
28      522     0   94061abc9409199061
29      540     0   94062abc9409299062
30      558     0   94063abc9409399063
31      576     0   94064abc9409499064
32      594     0   94065abc9409599065
33      612     0   94066abc9409699066
34      630     0   94067abc9409799067
35      648     0   94068abc9409899068
36      666     0   94069abc9409999069
37      684     0   94070abc9410099070
38      702     0   94071abc9410199071
39      720     0   94072abc9410299072
40      738     0   94073abc9410399073
41      756     0   94074abc9410499074
```

```
41      756     0   94074abc9410499074
42      774     0   94075abc9410599075
43      792     0   94076abc9410699076
44      810     0   94077abc9410799077
45      828     0   94078abc9410899078
46      846     0   94079abc9410999079
47      864     0   94080abc9411099080
48      882     0   94081abc9411199081
49      900     0   94082abc9411299082
50      918     0   94083abc9411399083
51      936     0   94084abc9411499084
52      954     0   94085abc9411599085
53      972     0   94086abc9411699086
54      990     0   94087abc9411799087
55      1008        0   94088abc9411899088
56      1026        0   94089abc9411999089
57      1044        0   94090abc9412099090
58      1062        0   94091abc9412199091
59      1080        0   94092abc9412299092
60      1098        0   94093abc9412399093
61      1116        0   94094abc9412499094
62      1134        0   94095abc9412599095
63      1152        0   94096abc9412699096
64      1170        0   94097abc9412799097
65      1188        0   94098abc9412899098
66      1206        0   94099abc9412999099
67      1224        0   94100abc9413099100
68      1242        0   94101abc9413199101
69      1260        0   94102abc9413299102
70      1278        0   94103abc9413399103
71      1296        0   94104abc9413499104
72      1314        0   94105abc9413599105
73      1332        0   94106abc9413699106
74      1350        0   94107abc9413799107
75      1368        0   94108abc9413899108
76      1386        0   94109abc9413999109
77      1404        0   94110abc9414099110
78      1422        0   94111abc9414199111
79      1440        0   94112abc9414299112
80      1458        0   94113abc9414399113
81      1476        0   94114abc9414499114
82      1494        0   94115abc9414599115
83      1512        0   94116abc9414699116
84      1530        0   94117abc9414799117
85      1548        0   94118abc9414899118
86      1566        0   94119abc9414999119
87      1584        0   94120abc9415099120
88      1602        0   94121abc9415199121
```

```
88      1602        0   94121abc9415199121
89      1620        0   94122abc9415299122
90      1638        0   94123abc9415399123
91      1656        0   94124abc9415499124
92      1674        0   94125abc9415599125
93      1692        0   94126abc9415699126
94      1710        0   94127abc9415799127
95      1728        0   94128abc9415899128
96      1746        0   94129abc9415999129
97      1764        0   94130abc9416099130
98      1782        0   94131abc9416199131
99      1800        0   94132abc9416299132
100     1818        0   94133abc9416399133
101     1836        0   94134abc9416499134
102     1854        0   94135abc9416599135
103     1872        0   94136abc9416699136
104     1890        0   94137abc9416799137
105     1908        0   94138abc9416899138
106     1926        0   94139abc9416999139
107     1944        0   94140abc9417099140
108     1962        0   94141abc9417199141
109     1980        0   94142abc9417299142
110     1998        0   94143abc9417399143
111     2016        0   94144abc9417499144
112     2034        0   94145abc9417599145
113     2052        0   94146abc9417699146
114     2070        0   94147abc9417799147
115     2088        0   94148abc9417899148
116     2106        0   94149abc9417999149
117     2124        0   94150abc9418099150
118     2142        0   94151abc9418199151
119     2160        0   94152abc9418299152
120     2178        0   94153abc9418399153
121     2196        0   94154abc9418499154
122     2214        0   94155abc9418599155
123     2232        0   94156abc9418699156
124     2250        0   94157abc9418799157
125     2268        0   94158abc9418899158
126     2286        0   94159abc9418999159
127     2304        0   94160abc9419099160
128     2322        0   94161abc9419199161
129     2340        0   94162abc9419299162
130     2358        0   94163abc9419399163
131     2376        0   94164abc9419499164
132     2394        0   94165abc9419599165
133     2412        0   94166abc9419699166
134     2430        0   94167abc9419799167
```

```
--------页头信息--------
页号：1555
页的空余空间：2835
该页当前存储的记录个数：37
偏移量表------记录内容
记录号    偏移量    是否删除
0       21      0   199963abc199993204963
1       42      0   199964abc199994204964
2       63      0   199965abc199995204965
3       84      0   199966abc199996204966
4       105     0   199967abc199997204967
5       126     0   199968abc199998204968
6       147     0   199969abc199999204969
7       168     0   199970abc200000204970
8       189     0   199971abc200001204971
9       210     0   199972abc200002204972
10      231     0   199973abc200003204973
11      252     0   199974abc200004204974
12      273     0   199975abc200005204975
13      294     0   199976abc200006204976
14      315     0   199977abc200007204977
15      336     0   199978abc200008204978
16      357     0   199979abc200009204979
17      378     0   199980abc200010204980
18      399     0   199981abc200011204981
19      420     0   199982abc200012204982
20      441     0   199983abc200013204983
21      462     0   199984abc200014204984
22      483     0   199985abc200015204985
23      504     0   199986abc200016204986
24      525     0   199987abc200017204987
25      546     0   199988abc200018204988
26      567     0   199989abc200019204989
27      588     0   199990abc200020204990
28      609     0   199991abc200021204991
29      630     0   199992abc200022204992
30      651     0   199993abc200023204993
31      672     0   199994abc200024204994
32      693     0   199995abc200025204995
33      714     0   199996abc200026204996
34      735     0   199997abc200027204997
35      756     0   199998abc200028204998
36      777     0   199999abc200029204999
**************************************************
```

七、 实验感想

首先，要对我们组没有及时验收实验一、且迟交实验报告的情况表示抱歉，由于组内分工等问题，导致实验进展一直不理想。虽然我们几个人能力和经验都比较欠缺，但态度上还是很积极的，每周都有进行小组讨论，在没能提交的这段时间里也一直尽心尽力地去思考、去设计、去写代码，所以希望老师和助教们能够谅解我们的延迟。

其次，本次实验从无到有，一点一滴，从中有很多收获，对数据库管理系统的存储层面有了更深刻的理解，把知识从书本上搬到实践中也极大地提高和锻炼了编程能力。