# 实验一实验报告

### 一、数据文件的组织

在我们实验中使用了页式组织方式。

系统文件：System.dbf 包含多个Datafile，这里的Datafile是数据文件，每个Datafile对应一个表，表名和datafile相同。

系统文件里面存的是系统的全局信息，包括Page Size， Buffer Size，Filenumber，Datafile等。

* Page Size：块的大小，系统默认使用8K。
* Buffer Size：缓冲区的大小，为块的整数倍，系统使用的是块大小的32倍。
* Filenumber：整个数据库所包含的数据文件个数，系统默认最大为100。
* Dadafile: 是一个结构体，包含文件的ID和Name。

系统文件的组织结构如图1所示。

每个Datafile的第一块为文件头，文件头包含PageNumber，FreePageNumber，Page的位图，使用为1，不使用为0。

* Pagenumber：当前文件的总块数。
* FreePageNumber：空闲块数。
* 每个Page包含头信息和数据信息。

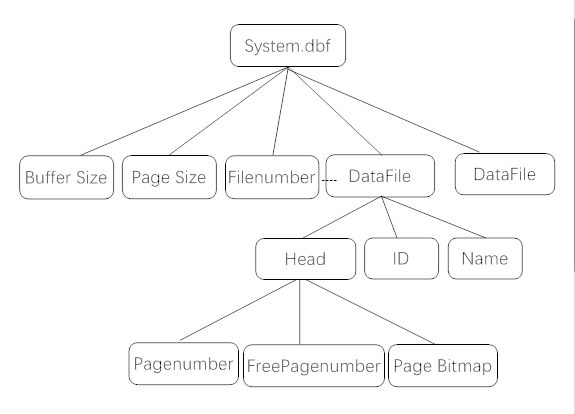


图1 系统文件的组织方式

### 二、缓冲区管理

缓冲区的页面是使用连续的内存块作为整个数据库的全局缓冲区。每块缓冲区的大小和数据文件的Page Size大小相等，也为8K。

当所使用的块数未达到总数时，直接分配一块新的Page。当达到的总数时，则需要做页面替换，缓冲区的替换方式为先进先出，用循环队列的方式实现。

循环队列的方式如图2所示：

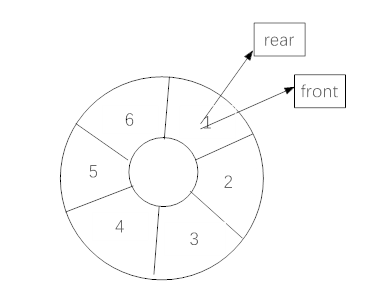


图2 缓冲区的替换方式，循环队列

### 三、空闲空间管理

这里是文件的空闲管理。一个DataFile包含多个Page，每个Page的使用情况在文件头中描述，包含 fileId， pageOfFile，freeCount，isPageFree[Max\_Page\_Per\_File]。

* fileId： 文件的ID号
* pageOfFile： 文件中包含的page数
* freeCount： 文件中可用的文件数
* isPageFree[Max\_Page\_Per\_File]： page的位图数组，只存储0或者1

表1 文件空闲块管理示意图

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| fileId=1, pageOfFile=21, freeCount=15, isPageFree=[11111000000............] | | | | | | |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |

### 四、代码及函数列表

**Structfile.h 文件结构头文件**

**Memblock.h 缓冲区管理头文件**

**Datafiles.c 文件管理源文件**

* isFileExist 函数 判断文件是否存在
* createSysFile 函数 创建系统文件
* saveSysFile 函数 保存系统文件
* readSysFile 函数 读系统文件
* createDataFile 函数 创建数据文件
* getFreePage 函数 获得空闲页
* writeNewPageToFile函数 往文件写新页
* readPageFromFile 函数 从文件读一页
* displayDBInfo 函数 显示数据库信息
* initDataFile 函数 初始化数据文件

**Buffmgr.c 缓冲区管理源文件**

* initQueue函数 初始化队列
* EnterQueue函数 入队
* DelQueue函数 出队
* allocateBuff函数 分配缓冲区
* queryFreeBuff函数 查询空闲缓存
* buffSwith 函数 缓冲区置换
* initMemory函数 初始化缓冲区