BufferManager 概要设计说明书

负责人：罗道锋
编写人：罗道锋
系统版本号：OrientX Version 1.5
完成时间：2003/12/16
开发单位：中国人民大学 IDKE 实验室 XML 工作组

1. 引言

本说明书详细说明了OrientX1.5中BufferManager的设计和实现。

2. 概述

BufferManager管理数据库的缓冲区。任何物理块都将被读到缓冲区进行读写，如果缓冲区已满，则采用LCR算法淘汰旧的缓冲区。

任何缓冲区块被请求读写时，该物理块的用户数加1；而该缓冲区被释放时，用户数减1。用户数为0的缓冲区叫做空闲缓冲区，该缓冲区可以被其它物理块占用（如果该缓冲区被改写过，需要先刷出到磁盘）；反之，如果缓冲区的用户数大于0，则该缓冲区不能被其它物理块占用。所以，在读写操作完成后，请释放缓冲区，以免造成缓冲区长期占用得不到释放。

3. 总体设计

BufferManager 对外提供的接口是 Buffer 类。提供的接口包括申请/释放缓冲区，读/写缓冲区，刷出缓冲区等。Buffer 类含两个内部类：BufferCtrl 和 IOCtrl。其中，BufferCtrl 类负责用 LCR 算法管理缓冲区的申请和释放；IOCtrl 类负责物理块 IO 读写。两个内部类对外都是隐藏的，用户只需要用到 Buffer 类提供的接口。

OrientX 中的物理块是定长的物理块，由参数 NX_FREEPAGE_CAPACITY 决定。所以，缓冲区也是定长的，缓冲区的定义是 static unsigned char bufPool[BufPoolSize][NX_FREEPAGE_CAPACITY]。其中，BufPoolSize 定义了缓冲区的个数。

4. 接口设计

Buffer类提供了六个接口，定义如下：

Buffer类接口设计	
函数定义	说明
short GetPage2Buf(int setID, int lpno,UserModes userMode);	把物理块读进缓冲区，返回缓冲区号
short FreeBuf(short bufNo);	释放缓冲区
short GetNewBuf (int setID, int lpNo,UserModes	为一个新的物理块分配一个缓冲区，由于

userMode);	该物理块没有内容，所以不用进行IO读写
short ReadBuf(short bufNo, short offset, short len, void *local);	读取缓冲区
short WriteBuf(short bufNo, short offset, short len, void *local);	写缓冲区
short FlushBuf(short bufNo);	把缓冲区强制刷出到磁盘

5. 数据结构设计

5.1 逻辑结构设计

5.1.1 BufferCtrl 类

BufferCtrl 类负责缓冲区的管理，包括缓冲区的申请/释放等。为了管理缓冲区，系统定义了一个缓冲区控制块 BufCtrlBlk 的结构体：

```
struct BufCtrlBlk{
    int setID;                  //memory page belonging to dataSet ID
    int lpNo;                   //logical page no
    short useCount;             // user count
    BufferStates bufSt;         //   buffer page status:   0: IOFree/ 1: Inputting/ 2: Outputting
    UserModes userMode;         //user status : 0--read only, 1--read and write
    UpdateLabels ifUpdate;      //   modify sign:   0--no modify   1--modified
    short hashNext;             //Hash queue backward pointer
    short hashPrev;             //Hash queue forward pointer,
                                //Hash queue include: FreeChain and UsedChain
    short freeNext;             //total Free queue backward pointer
    short freePrev;             // total free queue forward pointer
};
```

此外，BufferCtrl 还定义了如下数据成员：

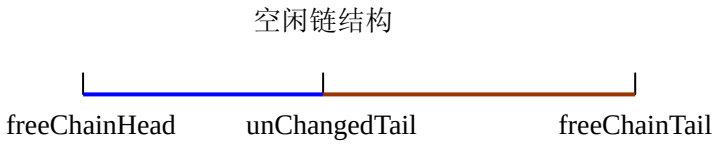
```
static BufCtrlBlk bufCtrlBlk[BufPoolSize];
static short HashUsedChainHead[HashLen];
static short HashFreeChainHead[HashLen];

static short freeChainHead;
static short unChangedTail;
static short freeChainTail;
static short freeNum;
```

BufCtrlBlk 定义了该缓冲区里面的物理块所属的 setID 和 lpNo，记录了缓冲区的状态（用户数和读写状态以及是否更新过），还维护了以下链表：

1、一条完整的空闲链（使用 freeNext 和 freePrev 指针）。

这条空闲链的前半部分是没有被更改过的缓冲区，后半部分是被更改过的缓冲区。这两段以 unChangedTail 作为分界线。freeChainHead 和 freeChainTail 分别是这条空闲链的链头和链尾。



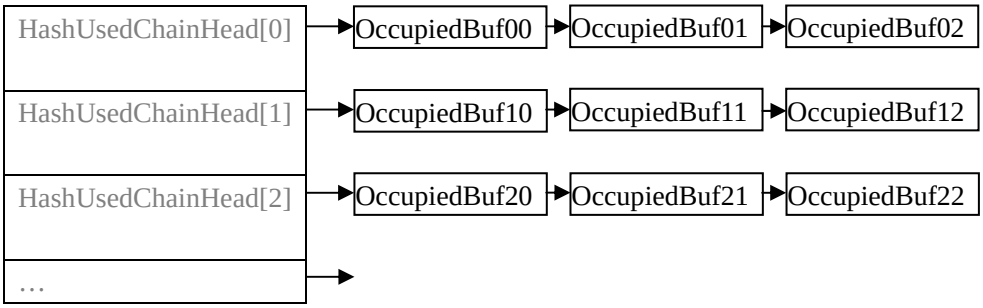
当一个缓冲区被释放而且用户数为 0（空闲时），如果该缓冲区被修改过，则链接到空闲链的尾部（即红色部分的尾部）；如果该缓冲区没有被修改过，则链接到未更改空闲链的尾部（即蓝色部分的尾部）。

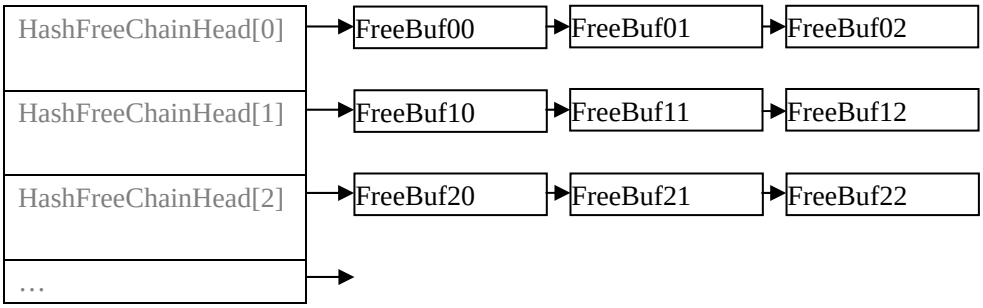
当申请缓冲区时，总是把 freeChainHead 分配出去。这样，除了实现 LCR 算法之外（最近最少使用的优先淘汰），而且使没有被更新过的缓冲区优先淘汰。因为没有更新过的缓冲区可以直接被覆盖，不用刷出到磁盘，因此可以减少一次 IO 写。

2、Hash 查找链

Hash 查找链是为了快速地查找某一个存储了某一个物理块的缓冲区。比如，当上层函数需要访问某一个物理块(SetID, LpNo)，我们需要首先把它读进缓冲区。但是该物理块可能原来已经被读进来过，可能是占用状态（用户数大于 0）或者空闲状态（用户数等于 0），我们就需要该物理块被读到那个缓冲区了。也就是说，我们需要一个从(SetID, LpNo)到 BufNo 的快速的映射。Hash 查找链就是为了建立这种映射。

Hash 查找链使用 hashNext 和 hashPrev 指针，查找链有两种：一种是占用查找链，另一种是空闲查找链。被占用的缓冲区将被放在占用查找链，空闲缓冲区将放在空闲查找链。它们如下图所示：





BufferCtrl类接口设计	
函数定义	说明
short GetFreeBuf();	释放一个缓冲区
short GetLpBufNo(int setID, int lpNo);	查找该物理块是否已经在缓冲区中
void Link2FreeChain(short bufNo);	把缓冲区连接到空闲链
void UnlinkFromFreeChain(short bufNo);	把缓冲区从空闲链摘下
void Link2UsedChain(short bufNo);	把缓冲区连接到占用查找链
void UnlinkFromUsedChain(short bufNo);	把缓冲区从占用查找链摘下
void FlushAll();	刷出所有缓冲区
int GetLpNo(short bufNo);	得到缓冲区保存的物理块号
void SetLpNo(short bufNo,int lpNo);	设置缓冲区保存的物理块号
int GetSetID(short bufNo);	得到缓冲区保存的物理块号所在的 SetID
void SetSetID(short bufNo, int setID);	设置缓冲区保存的物理块号所在的 SetID
BufferStates GetBufSt(short bufNo);	得到缓冲区的状态
void SetBufSt(short bufNo,BufferStates st);	设置缓冲区的状态
short GetUseCount(short bufNo);	得到缓冲区的用户数
void SetUseCount(short bufNo,short usecount);	设置缓冲区的用户数
UserModes GetUserMode(short bufNo);	得到缓冲区的读写模式
void SetUserMode(short bufNo,UserModes userMode);	设置缓冲区的读写模式
UpdateLabels GetIfUpdate(short bufNo);	得到缓冲区的是否被更新
void SetIfUpdate(short bufNo,UpdateLabels ifUpdate);	设置缓冲区的是否被更新

5.1.2 IOCtrl 类

IOCtrl 类负责缓冲区与磁盘之间的 IO 操作。目前，它使用带操作系统缓冲的读写。如果需要，可以改写 IOCtrl 的实现，使之不使用操作系统的 IO 缓冲。该类很简单，只有两个静态函数：

Buffer类接口设计	
函数定义	说明
static short FlushBuf(unsigned char *buffer,int setID, int lpNo);	刷出缓冲区到磁盘
static short Read2Buf(unsigned char *buffer, int setID, int lpNo);	从磁盘读取内容到缓冲区

6. 出错处理