Buffer Manager 设计报告 

> by 李易非

一、设计简介

Buffer Manager 是数据库与数据文件进行交互的中间层，所有数据库调用 / 更新文件的行为，都必须通过 BufferManager 来管理。这样可以尽量减少内存与磁盘的速度鸿沟，把一些较为常用的 Block 存入 Buffer 中，为了尽量减少 I / O 操作，应该把一些 Block 存在 Buffer 中，数据库需要访问速度更快；

我们采取了 **时钟算法** 作为块替换策略，时钟算法的效率接近块替换策略 **LRU算法** ，但是实现却简单很多，所以我选择了时钟算法作为Buffer的块替换策略；为了实现时钟算法，我们在Buffer内部实现了一个循环链表，循环链表的 Node 是 BufferNode 类，BufferNode 中含有一个 Block 实体，一个标记位，以及指向下一个 BufferNode 的指针；

BufferNode 具有的功能如下 (简化版)：

```
class BufferNode
{
    Block * block;                  // 实际的Block内容
    bool reference_bit;             // 标记位，来确定时钟算法替换的块
    BufferNode * next;              // 指向下一个BufferNode

    Block * get_block();            // 获得BufferNode中的块
    void set_block(Block * block);  // 重置BufferNode中的块

};
```

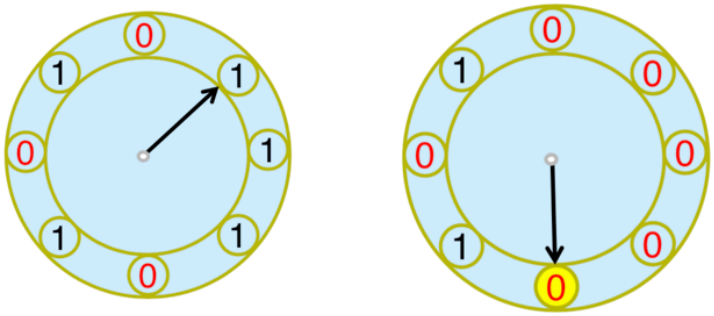
Buffer Manager 具有的功能如下 (简化版)：

```
class BufferManager
{
    const int max_block_count;      // Buffer 容量
    BufferNode * clock_ptr;         // ptr of the 时钟算法


    void WriteBlockBack(Block * block);                 // 写回一个 Block
    Block* getBlock(文件名, Block ID);                  // 通过文件名，Block
ID 获得对应块
    void addBlock(Block * block);                       // 在Buffer中加入一个
块
    void DeleteBlockByFile(文件名);                     // 删除所有与文件名相关
的块
    void WriteAllBack();                                // 写回Buffer里的所有
块

};
```

二、时钟算法介绍



Buffer中的每一个结点都有一个参考位，在需要替换Buffer中的块时，

- 从时钟算法指针开始遍历，若参考位为1，则将该节点参考位置为0，前往下一块；
- 重复上述过程直到找到一个参考位为0的节点，将其替换；

  每当有新块加入时，将其参考位变为1；

三、具体说明

1、成员变量说明

- `max_block_count`：Buffer容量，可以容纳的最大块数量；
- `clock_ptr`：时钟算法指针，用于遍历循环链表；

2、 成员函数说明

- `WriteBlockBack(Block * Block)`
  - 通过 Block 对象附带的信息：文件名，Block ID，计算 Block 应该写入的位置，写入文件对应位置；

```
OpenFile(Block.FileName);
写入的相对地址 = Block ID * 4096;
移动文件指针到相对地址;
写入 Block.Content;
```

- `Block* getBlock(文件名, Block ID)`
  - 通过文件名与Block ID 来得到对应的 Block
    - 若 Block 在 Buffer 中，则直接返回；
    - 若 Block 不再 Buffer 中，则从对应文件中读入；

```
if Block is already in Buffer
    找到Block并返回;
else
    从文件中读取 Block;
    加入 Buffer;
    返回读取的 Block;
```

- `void addBlock(Block * block)`
  - 把 Block 加入时钟算法的循环链表中，进行必要的替换；
  - 时钟算法指针开始遍历，若参考位为1，则将该节点参考位置为0，前往下一块；
  - 重复上述过程直到找到一个参考位为0的节点，将其替换；

```
while clock_pointer.refbit = 0
    clock_pointer = clock_pointer->next;

if clock_pointer.Block is dirty
    Write Block Back;

Replace the found Block by new Block;
```

- `void DeleteBlockByFile(文件名)`
  - 删除与文件名相关的所有 Block;
  - 遍历Buffer，找到相关的Block，删除即可；

```
for i = 1 : BufferSize
    if BufferNode.Block.FileNmae = 文件名
        if Block is dirty
            Write Block Back;
        Delete Block from Buffer;
    BufferNode = BufferNode.next;
```

- `void WriteAllBack()`
  - 把 Buffer 中所有脏块写入文件中；

```
for i = 1 : BufferSize
    if BufferNode.Block.FileNmae = 文件名
        if Block is dirty
            Write Block Back;
    BufferNode = BufferNode.next;
```