图文 14 当我们更新Buffer Pool中的数据时，flush链表有什么用?

当我们更新Buffer Pool中的数据时，flush链表有什么用?

如何提问：每篇文章都有评论区，大家可以尽情留言提问，我会逐一答疑
如何加群：购买狸猫技术窝专栏的小伙伴都可以加入狸猫技术交流群，一个非常纯粹的技术交流的地方

具体加群方式，请参见目录菜单下的文档：**《MySQL专栏付费用户如何加群》**（购买后可见）

1、昨日思考题解答

我们先解答一下昨日的思考题，昨天是问了大家一个问题，Buffer Pool中会不会有内存碎片?

答案是：当然有

因为Buffer Pool大小是你自己定的，很可能Buffer Pool划分完全部的缓存页和描述数据块之后，还剩一点点的内存，这一点点的内存放不下任何一个缓存页了，所以这点内存就只能放着不能用，这就是内存碎片。

那怎么减少内存碎片呢?

其实也很简单，数据库在Buffer Pool中划分缓存页的时候，会让所有的缓存页和描述数据块都紧密的挨在一起，这样尽可能减少内存浪费，就可以尽可能的减少内存碎片的产生了。

如果你的Buffer Pool里的缓存页是东一块西一块，那么必然导致缓存页的内存之间有很多内存空隙，这就会有大量的内存碎片了。

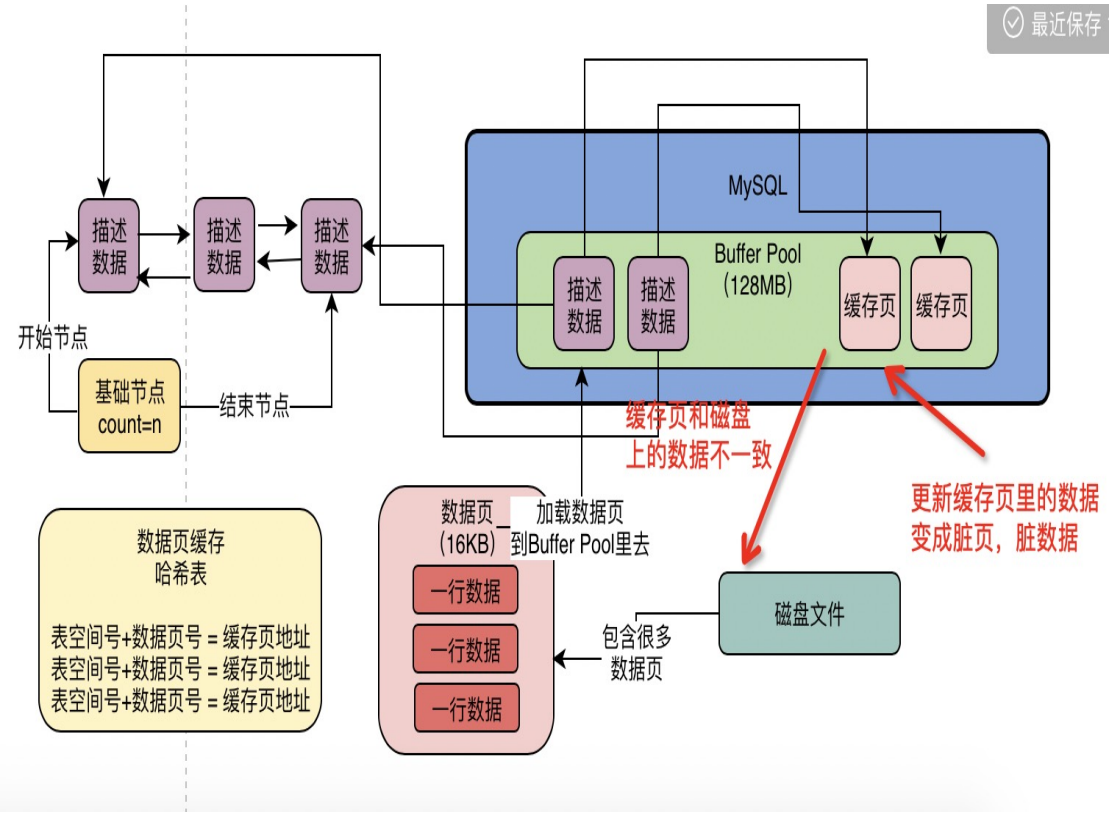
2、脏数据页到底为什么会脏?

接着我们看一个很关键的问题，你在执行增删改的时候，如果发现数据页没缓存，那么必然会基于free链表找到一个空闲的缓存页，然后读取到缓存页里去，但是如果已经缓存了，那么下一次就必然会直接使用缓存页。

反正不管怎么样，你要更新的数据页都会在Buffer Pool的缓存页里，供你在内存中直接执行增删改的操作。

接着你肯定会去更新Buffer Pool的缓存页中的数据，此时一旦你更新了缓存页中的数据，那么缓存页里的数据和磁盘上的数据页里的数据，是不是就不一致了?

这个时候，我们就说缓存页是脏数据，脏页



3、哪些缓存页是脏页呢?

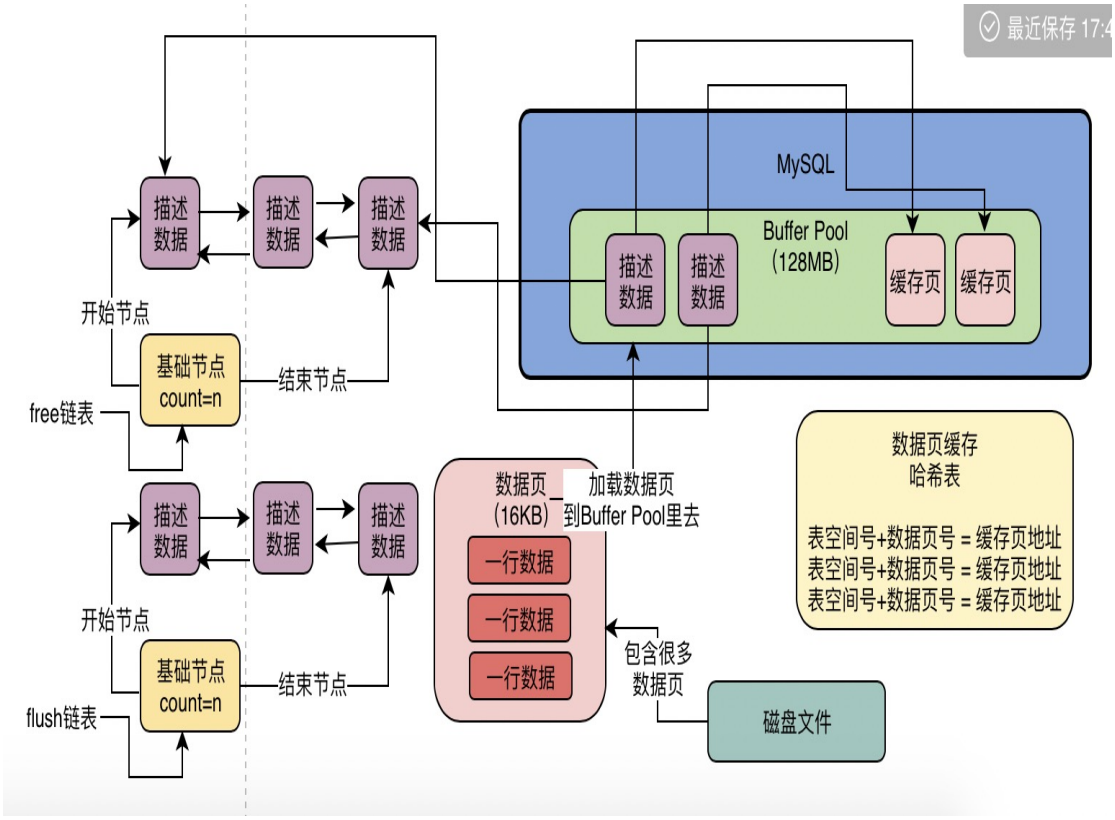
其实通过之前的学习，我们都是知道一点的，最终这些在内存里更新的脏页的数据，都是要被刷新回磁盘文件的。

但是这里就有一个问题了，不可能所有的缓存页都刷回磁盘的，因为有的缓存页可能是因为查询的时候被读取到Buffer Pool里去的，可能根本没修改过!

所以数据库在这里引入了另外一个跟free链表类似的**flush链表**，这个flush链表本质也是通过缓存页的描述数据块中的两个指针，让被修改过的缓存页的描述数据块，组成一个双向链表。

凡是被修改过的缓存页，都会把他的描述数据块加入到flush链表中去，flush的意思就是这些都是脏页，后续都是要flush刷新到磁盘上去的

所以flush链表的结构如下图所示，跟free链表几乎是一样的。



4、flush链表构造的伪代码演示

我们用一些伪代码来给大家展示一下这个flush链表的构造过程，比如现在缓存页01被修改了数据，那么他就是脏页了，此时就必须把他加入到flush链表中去

此时缓存页01的描述数据块假设如下所示

```
// 描述数据块
DescriptionDataBlock {
   // 这是缓存页01的数据块
   block_id = block01
   // 在free链表中的上一个节点和下一个节点
   // 因为这个缓存页已经被更新过了，肯定不在free链表里了
   // 所以他在free链表中的两个指针都是null
   free_pre = null
   free_next = null
   // 在flush链表中的上一个节点和下一个节点
   // 现在因为flush链表中就他一个节点，所以也都是null
   flush_pre = null
   flush_next = null
}
// flush链表的基础节点
FlushLinkListBaseNode {
    // 基础节点指向链表起始节点和结束节点的指针
    // flush链表中目前就一个缓存页01，所以指向他的描述数据块
    start = block01
    end = block01
    // flush链表中有几个节点
    count = 1
}
```

好了，我们可以看到，现在flush链表的基础节点就指向了一个block01的节点，接着比如缓存页02被更新了，他也是脏页了，此时他的描述数据块也要被加入到flush链表中去

此时伪代码如下：

```
// 描述数据块
DescriptionDataBlock {
   // 这是缓存页01的数据块
   block_id = block01
   // 在free链表中的上一个节点和下一个节点
   // 因为这个缓存页已经被更新过了，肯定不在free链表里了
   // 所以他在free链表中的两个指针都是null
   free_pre = null
   free_next = null
   // 在flush链表中的上一个节点和下一个节点
   // 现在因为flush链表中他是起始节点，所以他的flush_pre指针是null
   flush_pre = null
   // 然后flush链表中他的下一个节点是block02，所以flush_next指向block02
   flush_next = block02
}
// 描述数据块
DescriptionDataBlock {
   // 这是缓存页02的数据块
   block_id = block02
   // 在free链表中的上一个节点和下一个节点
   // 因为这个缓存页已经被更新过了，肯定不在free链表里了
   // 所以他在free链表中的两个指针都是null
   free_pre = null
   free_next = null
   // 在flush链表中的上一个节点和下一个节点
   // 现在因为flush链表中，他是尾节点，他的上一个节点是block01
   // 他的下一个节点就是null
   flush_pre = block01
   flush_next = null
}
// flush链表的基础节点
FlushLinkListBaseNode {
    // 基础节点指向链表起始节点和结束节点的指针
    // flush链表中目前有缓存页01和缓存页02，所以指向他们的描述数据块
    start = block01 // 起始节点是block01
    end = block02 // 尾巴节点是block02
    // flush链表中有几个节点
    count = 2
}
```

大家可以看到，当你更新缓存页的时候，通过变换缓存页中的描述数据块的flush链表的指针，就可以把脏页的描述数据块组成一个双向链表，也就是flush链表，而且flush链表的基础节点会指向起始节点和尾巴节点。

通过这个flush链表，就可以记录下来哪些缓存页是脏页了！

End