实验1：缓冲池管理器实现

邹兆年，哈尔滨工业大学计算学部，znzou@hit.edu.cn

https://gitee.com/HIT-DB/hit-db-class-rucbase-lab

一、实验目的

1. 掌握Rucbase缓冲池页面替换策略的实现方法。
2. 掌握Rucbase缓冲池管理器的实现方法。

二、相关知识

1. 缓冲池的组成
2. 缓冲池的页面替换策略
3. 缓冲池访问请求的处理方法

三、实验内容

本实验包括2项任务。

任务1：缓冲池页面替换策略实现

补全`LRUReplacer`类，实现最近最少使用（Least Recently Used, LRU）页面替换策略。`LRUReplacer`类继承了`Replacer`类。当缓冲池没有空闲页面时，缓冲池管理器需要使用`Replacer`类实现的页面替换策略选择一个页面进行淘汰。具体完成如下任务。

（1）阅读代码

阅读`src/replacer`目录下的代码。

- `src/replacer/replacer.h`
- `src/replacer/lru_replacer.h`
- `src/replacer/lru_replacer.cpp`

理解`LRUReplacer`类的设计，并回答下列问题：

1. `LRUlist_`的作用是什么？
2. `LRUhash_`的作用是什么？
3. `LRUlist_`和`LRUhash_`的关系是什么？

（2）实现`LRUReplacer::victim`函数

函数声明：

```
bool LRUReplacer::victim(frame_id_t *frame_id);
```

功能：获取待淘汰的页面。参数和返回值的含义参考代码注释。当缓冲池管理器需要淘汰页面时，它会调用该函数获得待淘汰页面所在帧的编号。

实现：参考代码注释。删除`LRUlist_`中最久被取消固定（unpin）的帧，然后通过参数`frame_id`返回该帧的编号。

（4）实现`LRUReplacer::pin`函数

函数声明：

```
void LRUReplacer::pin(frame_id_t frame_id);
```

功能：固定（pin）页面。参数和返回值的含义参考代码注释。当缓冲池管理器要固定帧`frame_id`中的页面时，调用该函数。

实现：参考代码注释。如果`LRUlist_`中包含`frame_id`，则从`LRUlist_`中删除`frame_id`。

（5）实现`LRUReplacer::unpin`函数

函数声明：

```
void LRUReplacer::unpin(frame_id_t frame_id);
```

功能：取消固定（unpin）页面。参数和返回值的含义参考代码注释。当缓冲池管理器要取消固定（unpin）帧`frame_id`中的页面时（即该页面的`pin_count`变为0时），调用该函数。

实现：参考代码注释。将`frame_id`插入到`LRUlist_`中，放在最近被取消固定（unpin）的帧所处的位置。

（6）单元测试

单元测试代码在文件`src/test/storage/lru_replacer_test.cpp`中。

执行下列命令，进行单元测试。

```
cd build
make lru_replacer_test
./bin/lru_replacer_test
```

（7）注意事项

不允许修改任何公有函数的声明。

任务2：缓冲池管理器实现

补全`BufferPoolManager`类，实现Rucbase缓冲池管理器，具体完成如下任务。

（1）阅读代码

阅读下列文件中的代码。

- `src/storage/page.h`
- `src/storage/buffer_pool_manager.h`
- `src/storage/buffer_pool_manager.cpp`

理解`Page`和`BufferPoolManager`类的设计，并回答下列问题：

1. `Page::is_dirty_`的作用是什么？
2. `Page::pin_count_`的作用是什么？
3. `BufferPoolManager::page_table_`的作用是什么？
4. `BufferPoolManager::free_list_`的作用是什么？

（2）实现`BufferPoolManager::find_victim_page`函数

函数声明：

```
bool BufferPoolManager::find_victim_page(frame_id_t* frame_id);
```

功能：寻找被淘汰的页。参数和返回值的含义参考代码注释。

- 如果找到了被淘汰的页，则将`*frame_id`赋值为该页的编号，并返回`true`。
- 如果未找到被淘汰的页，则将`frame_id`赋值为`null`，并返回`false`。

实现：参考代码注释。考虑两种情况：

- 情况1：缓冲池中有空闲帧（`free_list_`不为空）。从`free_list_`取出第一个空闲帧的编号，并从`free_list_`中移除。
- 情况2：缓冲池中无空闲帧（`free_list_`为空）。调用`replacer_->victim()`，返回最适合被替换的帧的编号。

（3）实现`BufferPoolManager::fetch_page`函数

函数声明：

```
Page* BufferPoolManager::fetch_page(PageId page_id);
```

功能：获取缓冲池中的指定页面。参数和返回值的含义参考代码注释。

实现：参考代码注释。内部实现逻辑包括更新页表和页面、固定页面、寻找淘汰页等。如果缓冲池中不存在该页面，需要用`DiskManager`从磁盘中读取。需要用到之前实现的`Replacer::pin`、`Replacer::victim`、`DiskManager::read_page`等函数。

（4）实现`BufferPoolManager::unpin_page`函数

函数声明：

```
bool BufferPoolManager::unpin_page(PageId page_id, bool is_dirty);
```

功能：使用完页面后，对该页面取消固定。参数和返回值的含义参考代码注释。

实现：参考代码注释。

（5）实现`BufferPoolManager::flush_page`函数

函数声明：

```
bool BufferPoolManager::flush_page(PageId page_id);
```

功能：将缓冲池中指定页面强制刷写到磁盘。“强制”是指无论该页的引用次数是否大于0，无论该页是否为脏页，都将其刷新到磁盘。参数和返回值的含义参考代码注释。

实现：参考代码注释。

（6）实现`BufferPoolManager::new_page`函数

函数声明：

```
Page* BufferPoolManager::new_page(PageId* page_id);
```

功能：在缓冲池中申请创建一个新页面。参数和返回值的含义参考代码注释。

实现：参考代码注释。基本实现逻辑包括更新页表和页面、固定页面、寻找淘汰页等。此外，需要用`DiskManager`分配页面编号，并返回这个新页面的编号。提示：需要用到之前实现的`Replacer::pin`、`Replacer::victim`、`DiskManager::allocate_page`等函数。

（7）实现`BufferPoolManager::delete_page`函数

函数声明：

```
bool BufferPoolManager::delete_page(PageId page_id);
```

功能：删除指定页面。参数和返回值的含义参考代码注释。

实现：参考代码注释。基本实现逻辑包括更新页表和页面、更新空闲帧列表等。注意：只有引用次数（pin count）为0的页面才能被删除。

（8）实现`BufferPoolManager::flush_all_pages`函数

函数声明：

```
void BufferPoolManager::flush_all_pages(int fd);
```

功能：将缓冲池中属于指定文件的所有页面都刷写到磁盘。参数和返回值的含义参考代码注释。

实现：参考代码注释。建议不要通过调用`BufferPoolManager::flush_page`函数来实现，因为容易导致死锁。

（9）单元测试

单元测试代码在文件`src/test/storage/buffer_pool_manager_test.cpp`中。

执行下列命令，进行单元测试。

```
cd build
make buffer_pool_manager_test
./bin/buffer_pool_manager_test
```

（10）注意事项

- 不允许修改任何公有函数的声明。
- 不要实现和使用`BufferPoolManager::update_page`函数。

四、考核方法

1. 实验完成情况（80%）。根据单元测试通过情况和代码理解情况给分。
    - 任务1：缓冲池页面替换策略实现（20%）
    - 任务2：缓冲池管理器实现（60%）
2. 实验报告（20%）。根据实验报告的完整性、科学性和规范性评分。