MiniSQL阶段报告1

——Disk and Buffer Pool Manager

> 第七小组

1.1 实验概述

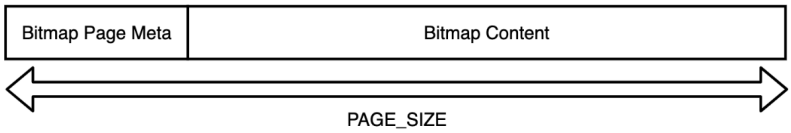
本阶段我们需要实现Disk Manager和Buffer Pool Manager模块

- Disk Manager主要负责数据库文件中数据页的分配和回收（Bitmap实现），以及数据页中数据的读取和写入。
- Buffer Pool Manager将磁盘中的数据页从内存中移动到磁盘。

1.2 实现一个简单的位图页

位图页（Bitmap page）

- 占用 `PAGE_SIZE` （4KB）的空间，标记一段连续页的分配情况。
- 组成：



  - 元信息（Bitmap Page Meta）
    - 已经分配的页的数量（ `page_allocated_` ）
    - 下一个空闲的数据页(`next_free_page_`)
  - Bitmap存储的具体数据

实现

- `BitmapPage::AllocatePage(&page_offset)` 分配一个空闲页
  - 判断是否已经达到最大的supported size，如果已经达到，不能成功分配，返回 `false`
  - `page_offset = next_free_page_` ，通过 `byte_index value of page_offset / 8` 以及 `bit_index value of page_offset % 8` ，找到该页所在的位置，并且将该位置置1，更新已分配的页数 `page_allocated_` 以及 `next_free_page_`
- `BitmapPage::DeAllocatePage(page_offset)` 回收已经被分配的页
  - 通过 `page_offset` 计算bit位
  - 将该bit位置0
- `BitmapPage::IsPageFree(page_offset)` 判断给定的页是否是空闲的
  - 通过 `page_offset` 计算bit位
  - 如果该bit位为0，则空闲，返回 `true` ，否则返回 `false`

1.3 磁盘数据页管理

实现:

使用 `DiskManager` 来管理磁盘的页分配和释放。使用数据 `Disk Meta Page` 来管理维护各个分区 `extent` .各个分区由各自的 `bitmap` 维护分区内的页分配和释放。下面实现 `DiskManager` 这几个函数：

- `DiskManager::AllocatePage()`
  - 获取 `logical_page_id` 所在的分区的 `bitmap` ,调用此 `bitmap` 的 `AllocatePage` 函数，完成页的分配。
- `DiskManager::DeAllocatePage(logical_page_id)`

    - 获取 `logical_page_id` 所在的分区的 `bitmap` ,调用此 `bitmap` 的 `DeAllocatePage` 函数，完成页的释放。
- `DiskManager::IsPageFree(logical_page_id)`
    - 获取 `logical_page_id` 所在的分区的 `bitmap` ,调用此 `bitmap` 的 `IsPageFree` 函数，判断此页是否空闲。
- `DiskManager::MapPageId(logical_page_id)`
    - 根据一个分区的大小构造逻辑页号和物理页号的映射。

测试：

```
jsc@LAPTOP-J4D78JQ7:/mnt/d/ZJU_course/minisql_github/build$ ./test/disk_manager_test
[==========] Running 2 tests from 1 test suite.
[----------] Global test environment set-up.
[----------] 2 tests from DiskManagerTest
[ RUN      ] DiskManagerTest.BitMapPageTest
[       OK ] DiskManagerTest.BitMapPageTest (11 ms)
[ RUN      ] DiskManagerTest.FreePageAllocationTest
[       OK ] DiskManagerTest.FreePageAllocationTest (1692 ms)
[----------] 2 tests from DiskManagerTest (1705 ms total)

[----------] Global test environment tear-down
[==========] 2 tests from 1 test suite ran. (1709 ms total)
[  PASSED  ] 2 tests.
jsc@LAPTOP-J4D78JQ7:/mnt/d/ZJU_course/minisql_github/build$
```

1.4 LRU替换策略

实现：

将在buffer pool中没有pin的页帧数存放在一个链表 `lru_list_` 中，最新访问的页移到链表末尾，这样，链表第一个元素即最近最少访问的元素，根据LRU策略，即可以将它替换。

- `LRUReplacer::Victim(*frame_id)`
    - 找到最近最少访问的元素，即链表的起始元素，这是被victim的页面
    - 如果链表为空，返回 `false` ，此时在buffer pool中的页面都不可被替换
- `LRUReplacer::Pin(frame_id)`
    - 将该数据页从 `lru_list_` 中移除
- `LRUReplacer::Unpin(frame_id)`
    - 将该数据页放入 `lru_list_` 中
- `LRUReplacer::Size()` 即 `lru_list_` 链表元素的个数

测试：

```
[ 16%] Built target glogbase
[ 19%] Built target glog
[ 23%] Built target gtest
[ 28%] Built target minisql_test_main
Consolidate compiler generated dependencies of target minisql_shared
[ 30%] Building CXX object bin/CMakeFiles/minisql_shared.dir/buffer/lru_replacer.cpp.o
[ 33%] Linking CXX shared library libminisql_shared.so
[ 95%] Built target minisql_shared
Consolidate compiler generated dependencies of target lru_replacer_test
[ 97%] Linking CXX executable lru_replacer_test
[100%] Built target lru_replacer_test
[==========] Running 1 test from 1 test suite.
[----------] Global test environment set-up.
[----------] 1 test from LRUReplacerTest
[ RUN      ] LRUReplacerTest.SampleTest
[       OK ] LRUReplacerTest.SampleTest (0 ms)
[----------] 1 test from LRUReplacerTest (0 ms total)

[----------] Global test environment tear-down
[==========] 1 test from 1 test suite ran. (0 ms total)
[  PASSED  ] 1 test.
```

1.5 缓冲池管理

实现：

Buffer pool manager 从内存中找到要求的数据页，如果数据页不在内存中，从磁盘中获取相应数据页，并根据1.4实现的LRU规则将数据页储存到内存中，必要时将内存中的脏页面写入磁盘。同时该manager需要能够实现分配新数据页和释放数据页的功能。

- `BufferPoolManager::FetchPage(page_id)`
  - 根据逻辑页号获取对应的数据页，判断是否在内存中，如果在，直接返回指向该数据页的指针；
  - 如果不在，则需要从磁盘中寻找到数据，并且利用LRU规则从空闲表（先）和替换表中寻找写入该数据的内存位置；
  - 更新相关参数
- `BufferPoolManager::NewPage(&page_id)`
  - 分配一个新的数据页，并将它们写入磁盘和内存
- `BufferPoolManager::UnpinPage(page_id, is_dirty)`
  - 取消固定固定一个数据页，将数据页放入替换表中
- `BufferPoolManager::FlushPage(page_id)`
  - 将数据页存储到磁盘中，内存中这一块内容清空并将它放入到空闲表中
- `BufferPoolManager::DeletePage(page_id)`
  - 释放一个数据页，将其从内存和磁盘中都删除
- `BufferPoolManager::FlushAllPages()`
  - 将所有页面都存储到磁盘中

测试：

```
chenyu@LAPTOP-45N5IQPH:/mnt/c/Users/17260/Documents/GitHub/DB/build$ ./test/buffer_pool_manager_test
[==========] Running 1 test from 1 test suite.
[----------] Global test environment set-up.
[----------] 1 test from BufferPoolManagerTest
[ RUN      ] BufferPoolManagerTest.BinaryDataTest
110[       OK ] BufferPoolManagerTest.BinaryDataTest (96 ms)
[----------] 1 test from BufferPoolManagerTest (96 ms total)

[----------] Global test environment tear-down
[==========] 1 test from 1 test suite ran. (96 ms total)
[  PASSED  ] 1 test.
```