# CMU 15-445

## Lab1 Buffer Pool Manager

Task1 LRU

采用hash + 链表来表示需可以驱逐的页标号（frame\_id）

1. Pin函数从驱逐列表从删除对应的页
2. Unpin会将页表放入可驱逐链表，如果链表满了，则先驱逐一部分
3. Victim则会从链表末尾选择一个驱逐

Task2 Buffer Pool Manager

page\_id表示页的id，fram\_id表示buffer pool中的编号。

page\_table是page\_id和frame\_id的映射，free\_list存储具体的空闲的frame\_id。每次需要空闲的位置就从free\_list中寻找，如果需要寻找内存中的数据则从Page\_table中寻找。page有pin\_count还有is\_dirty标志

1. FetchPage，判断page\_table中是否存在这个页，如果有直接返回，没有就从free\_list中找一个空闲的frame，映射对应的值，最会从磁盘中获取这个页，初始化pin\_count后就返回。
2. UnpinPage，从page\_table中找到页， pin\_count减一，如果现在值为0，则调用Unpin，放入驱逐列表中。根据输入的is\_dirty修改脏页标记
3. FlushPage，根据page\_id找到buffer\_pool中的数据刷会磁盘
4. NewPage，根据page\_id建立page\_table的映射
5. DeletePage，主要是判断pin\_count的值，如果是0才执行。

## Lab2 B\_PLUS\_TREE

b\_plus\_tree:存储root\_page\_id、buffer\_pool\_manager、mutex

leaf\_page和internal\_page都继承自b\_plus\_tree\_page都包含了自己的page\_id还有parent\_page\_id

GetValue: 调用FIndLeadPage，然后在LeafPage中查找对应的值。

FindLeafPage，一路从root查找到底部，如果是read，则读到下一页就释放上一页的读锁，如果是write，则还需要判断是否安全，如果安全则会把一路存在栈中的页面全部释放掉锁.

安全主要是指是否会涉及到父页面的修改，如果是insert，页面太满就可能分裂，则对parent来说是不安全的，不能释放锁。如果是delete，则页面太少就可能合并，也会修改父页面，则是不安全的。

Insert: 首先判断是否有根节点，如果没有需要创建，然后调用FindLeafPage，获取叶子节点后找到位置插入，插入后判断有没有满，如果满了需要Split分裂，将分裂出的新节点InsertIntoParent,如果父节点也插满了，则需要分裂。根节点插满需要特殊判断

Remove类似，则是需要判断是否要合并。

乐观锁：从上到下都是读锁，只有最后修改叶子节点的时候才是写锁，如果写完发现没有分裂则直接成功。如果发现不能只修改叶子节点，则需要从头开始改成全程读锁

需要注意的点：

1. 加锁是先fetch页，然后latch。释放锁是Unlatch然后Unpin，不能交换顺序，如果先Unpin这个时候页面可能被直接回收掉了，再unlatch就会出错
2. 忘记Unpin会导致页面一直占用内存
3. root页面需要加锁，因为当一个线程一直等待root节点的时候，另一个线程此时已经分裂出一个新的root节点，上一个线程持有的root就过期了
4. sibling节点也要加锁，因为可能存在调用了从root出发访问到sibling但父节点已经释放了，所以后来的节点也可以访问到同一层。基本上访问任何节点都需要加锁，包括Split也需要，因为可能iterator
5. 可能出现死锁：iterator请求访问下一节点，下一个节点请求访问sibling节点

Lock和Latch的区别：

Lock是高级别的逻辑原语，保护的是数据库的内容，直接回滚

Latch是低级别的保护源语，保护的是线程的数据结构，支持回滚。

## Lab3 QUERY EXECUTION

ExecutorContext：包含了catalog（存储表相关的信息）、BufferPoolManager、LockManager、TransactionManager等场景需要的类

PlanNode（执行计划）：包括vector组成的数据、比较器、表信息等

ValueFactory封装了常用的值类型，用来做数据库的值类型

Schema：封装了列的类型相关的信息

Tablemetadata：包括schema存储列的数量和类型、tableheap存储真正的列值

IndexInfo：存储的是索引的信息，包括key\_schema、name、Index等

Expression：predicate就是tuple必须满足相应的谓词要求才能返回。分为comparsion、column\_value、constant\_value，比较的时候就是用不同类型的expression进行比较。

## Lab4 CONCURRENCY CONTROL

主要是循环判断是否有条件获取锁，没有就wait，可以授予锁

LockRequestQueue：针对的RID，将所有锁都串成链表，授予了就granted，执行完成就从list中删除。还有upgrading表示当前行的2PL状态

ShardLockSet和ExeclusiveLockSet由事务管理，存储的是当前事务对哪些RID进行了请求

Unclock：unlock后需要p判断队列情况从队列中唤醒一个或者所有的线程。让他们争夺锁

RunCycleDetection：遍历所有请求队列，同一个RID，排在后面的事务就在等待前面的，生成对应的等待图。运行DFS，DFS会用一个栈来表示当前访问的所有事务，栈反应了dfs的路径。最后根据DFS找到的事务，将其状态设置为abort。然后唤醒这个事务的请求锁，如果请求锁发现自己已经被aborted了，则这个事务的获取锁的过程会被抛出异常