Lab 2 Write-Up

Jordi Burbano (204 076 325), Keisuke Daimon (604 547 017)

February 17, 2015

I. DESIGN DECISIONS

The BufferPool class uses an LRU eviction policy. This is implemented using a LinkedList<Integer> of the hash codes that are the keys to the cachedPages hash map (which maps int hash codes to Page objects); these hash codes are listed from the least recently used page at the head to the most recently used page at the end. Updating the linked list occurs when a page is referenced in BufferPool.getPage() so that it moves to the end of the linked list as the most recently page. It also occurs during BufferPool.evictPage() when the evicted page's hash code is removed from the linked list. BufferPool.evictPage() simply refers to the hash code at the head of the linked list to remove the corresponding page from the buffer pool's hash map.

II. CHANGES TO THE API

We made no changes to the API.

III. MISSING OR INCOMPLETE ELEMENTS OF OUR CODE

For the methods getJoinField1Name() and getJoinField2Name() in Join.java, we are able to access a table's name by consulting Catalog.getTableName(). However, we did not know how to access the alias name as described by the requirement that field names should be quantified by alias or table name.

IV. LOGISTICS

We spent approximately 25 man-hours on the project.

We spent a significant amount of time trying to pass the EvictionTest system test. First, we discovered an issue attributable to not closing RandomAccessFile objects when reading or writing pages. Afterwards, much time was spent on overcoming an error that stated that 80 MB of RAM were being used when the limit was 5 MB. This error was related to our previous implementation of TransactionFileDbIterator, which stored all the pages in an ArrayList<Page>. To consume less memory, TransactionFileDbIterator now stores an ArrayList<PageId>; this change reduced the consumption of RAM to 1 MB.