ΠΑΝΕΠΙΣΤΗΜΙΟ ΚΥΠΡΟΥ

Τμήμα Πληροφορικής

ΕΠΛ 646 – Προχωρημένα Θέματα Βάσεων Δεδομένων

ΑΣΚΗΣΗ 1 – Υλοποίηση Διαχειριστή Κρυφής Μνήμης (Buffer Manager) της Βάσης Δεδομένων Minibase

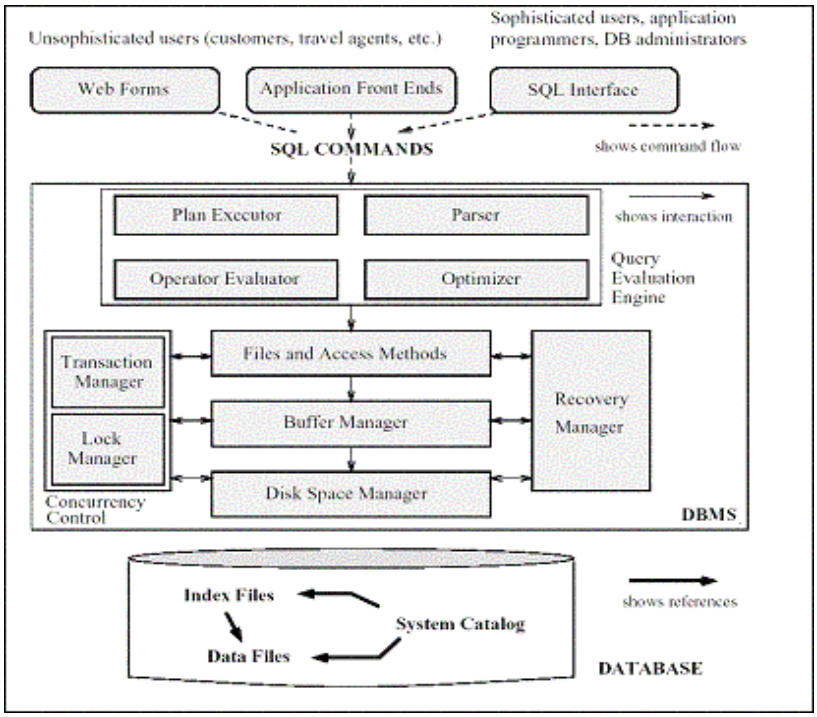
Διδάσκων: Δημήτρης Ζεϊναλιπούρ

Ημερομηνία Ανάθεσης: Δευτέρα, 14/9/2015
Ημερομηνία Παράδοσης: Πέμπτη, 3/10/2015 (18 μέρες)

(Να υποβληθεί ο κώδικας της κλάσης **BufMgr** μαζί με **όλα** τα πηγαία αρχεία στο Moodle)

Ι. Στόχος Άσκησης

Σε αυτή τη φάση πρέπει να υλοποιήσετε το στρώμα του διαχειριστή προσωρινής μνήμης (buffer management) (χωρίς υποστήριξη έλεγχου ταυτοχρονίας και επαναφοράς - concurrency control and recovery) για το σύστημα βάσεων δεδομένων Minibase. Ο κώδικας για το υπόστρωμα διαχειριστή χώρου δίσκου (Disk Space Manager) θα σας δοθεί. Συγκεκριμένα, πρέπει να μελετήσετε την περιγραφή της κατηγορίας DB, την οποία και θα χρησιμοποιήσετε εκτενώς σε αυτή την εργασία.



II. Προεργασία

1. Μελετήστε την περιγραφή του διαχειριστή προσωρινής μνήμης (Buffer Manager) από το βιβλίο (κεφάλαιο 9.4 του Ramakrishnan & Gehrke) καθώς επίσης τις σημειώσεις του μαθήματος (μέχρι και τη διάλεξη 4).
2. Μελετήστε το εκπαιδευτικό εγχειρίδιο για τη Minibase (minibase.pdf) το οποίο βρίσκεται συνημμένο μαζί με την εργασία. Το εγχειρίδιο θα σας βοηθήσει να καταλάβετε πως δουλεύει ο χειρισμός λαθών (error handling) και ο διαχειριστής αποθήκευσης (Storage Manager) στη Minibase.

III. Μεταγλώττιση και Εκτέλεση

Κατεβάστε το αρχείο **as1-bm.tar.gz** στο λογαριασμό σας στον κατάλογο project και αποσυμπιέστε το δίνοντας τις ακόλουθες εντολές:

1. mkdir project
2. cd project
3. tar -zxvf as1-bm.tar.gz
4. cd as1-bm/

Μετά από αυτό το βήμα πρέπει να έχετε τους ακόλουθους καταλόγους::

1. include/
2. **src/**

Τώρα μπορείτε να καλέσετε την εντολή **make** η οποία θα δημιουργήσει το εκτελέσιμο αρχείο **buftest**. Εάν δοκιμάσετε να εκτελέσετε το εν λόγω αρχείο θα σας επιστραφεί το μήνυμα ***Segmentation Fault***. Προφανώς για να δουλέψει σωστά το πρόγραμμα σας πρέπει να συμπληρώσετε τον κώδικα που λείπει από την κατηγορία **BufMgr** (και ότι άλλο χρειαστείτε για την υλοποίηση σας). Οι μεθόδοι ορίζονται (χωρίς το σώμα τους) στο αρχείο ***buf.C***. Το αρχείο ***correct_output*** περιέχει παράδειγμα επιτυχημένης εκτέλεσης.

IV. Διασύνδεση Διαχειριστή Κρυφής Μνήμης (Buffer Manager Interface)

Το απλοποιημένο buffer manager interface που θα υλοποιήσετε επιτρέπει σε ένα πρόγραμμα ανώτερου επιπέδου (higher level program) να δεσμεύσει και αποδεσμεύσει σελίδες στο δίσκο, να ανακτήσει μια σελίδα από το δίσκο στο διαχειριστή προσωρινής μνήμης και να το επικολλήσει (pin it), και να αποκολλήσει μια σελίδα από την προσωρινή μνήμη.

Οι μέθοδοι που πρέπει να υλοποιήσετε περιγράφονται ακολούθως:

```
class BufMgr {
public:
// This is made public just because we need it in your
// driver test.C . It could be private for real use.
Page* bufPool;
// The physical buffer pool of pages.

public:
    BufMgr(int numbuf, Replacer *replacer = 0);
    // Allocate "numbuf" pages (frames) for the pool in main memory.

   ~BufMgr();
   // Should flush all dirty pages in the pool to
   // disk before shutting down and deallocate the
   // buffer pool in main memory.

   Status pinPage(PageId PageId_in_a_DB, Page*& page,int emptyPage=0);
   // Check if this page is in buffer pool. If it is, increment the
   // pin_count and return a pointer to this page. If the pin_count was
   // 0 before the call, the page was a replacement candidate, but is no
   // longer a candidate
   // If the page is not in the pool, choose a frame (from the set of
   // replacement candidates) to hold this page, read the page (using
   // the appropriate DB class method) and pin it.
   // Also, must write out the old page in chosen frame if it is dirty
   // before reading new page. (You can assume that emptyPage == 0 for
   // this assignment.)

   Status unpinPage(PageId globalPageId_in_a_DB, int dirty, int hate);
   // hate should be TRUE if the page is ?hated? and FALSE otherwise.
   // Should be called with dirty==TRUE if the client has
   // modified the page. If so, this call should set the dirty bit
   // for this frame. Further, if pin_count>0 should decrement it.
   // If pin_count=0 before this call, return error.

   Status newPage(PageId& firstPageId, Page*& firstpage,int howmany=1);
   // Find a frame in the buffer pool for the first page
   // If a frame exists, call DB object to allocate a run of new pages // and pin it. (This call
   // allows a client of the Buffer Manager to allocate pages on disk.) If buffer is full, i.e., you
   // can't find a frame for the first page, return error.

  Status freePage(PageId globalPageId);
  // This method should be called to delete a page that is on disk.
  // This routine must call the DB class to deallocate the page.

  Status flushPage(int pageid);
  // Used to flush a particular page of the buffer pool to disk
  // Should call the write_page method of the DB class

  Status flushAllPages();
  // Flush all pages of the buffer pool to disk
};
```

Πολιτική αντικατάστασης Love/Hate

Θεωρητικά, η καλύτερη υποψήφια σελίδα για αντικατάσταση είναι η σελίδα που θα ζητηθεί τελευταία στο μέλλον. Εφόσον η υλοποίηση μιας τέτοιας πολιτική απαιτεί έναν μελλοντικό χρησμό, όλες οι πολιτικές αντικατάστασης προσπαθούν να δώσουν μια καλή προσέγγιση για αυτό. Η LRU πολιτική, παραδείγματος χάριν, χρησιμοποιεί το προηγούμενο μοτίβο πρόσβασης (access pattern) ως ένδειξη για το μέλλον. Εντούτοις, η διαδοχική ανάκληση σελίδων μπορεί να καταστρέψει αυτό το σχέδιο και η MRU γίνεται πιο κατάλληλη σε αυτήν την περίπτωση. **Σε αυτήν την εργασία πρέπει να εφαρμόσετε την πολιτική αντικατάστασης αγάπης/μίσους (Love/Hate).** Η πολιτική προσπαθεί να ενισχύσει την πρόβλεψη του μέλλοντος στηριζόμενη σε έναν υπαινιγμό (Hint) από τα ανώτερα επίπεδα για τη σελίδα. Το ανώτερο επίπεδο υπαινίσσεται στο διαχειριστή προσωρινής μνήμης ότι η σελίδα είναι αγαπημένη εάν είναι πιθανότερο ότι η σελίδα θα απαιτηθεί στο μέλλον, και μισητή εάν είναι πιθανότερο ότι η σελίδα δεν θα απαιτηθεί. **Αν πρέπει να αντικατασταθεί κάποια σελίδα, ο διαχειριστής προσωρινής μνήμης διαλέγει σελίδες από τις μισητές σελίδες και αν δεν βρει καμιά τότε από τις αγαπημένες.**

Ένα πρόβλημα που μπορεί να δημιουργηθεί είναι όταν μια σελίδα δηλωθεί αγαπημένη από ένα χρήστη και μισητή από κάποιον άλλο. Σε αυτή την περίπτωση υπερισχύει ο χαρακτηρισμό αγαπημένη.

Επισκόπηση Σχεδίασης και Υλοποίησης

Το *buffer pool* είναι μια συλλογή από *frames* (σειρά bytes της κύριας μνήμης μεγέθους σελίδας) την οποία την διαχειρίζεται ο διαχειριστής προσωρινής μνήμης. Πρέπει να αποθηκεύεται σαν ένας πίνακας **bufPool[numbuf]** αντικειμένων Page. Επιπρόσθετα, πρέπει να έχετε και ένα πίνακα **bufDescr[numbuf]** από *descriptors*, ένα για κάθε frame. Κάθε descriptor είναι μια εγγραφή με τα ακόλουθα πεδία:

Frame_Descriptor = (page number, pin_count, dirtybit)

Το πεδίο ***page number*** είναι ένα αντικείμενο **PageId**, το ***pin_count*** field είναι ένας integer και το ***dirtybit*** είναι boolean. Αυτή η εγγραφή περιγράφει τη σελίδα που είναι αποθηκευμένη στο αντίστοιχο frame. Μια σελίδα χαρακτηρίζεται από ένα ***page number*** το οποίο δημιουργείται από την κατηγορία DB όταν αυτή η σελίδα δεσμεύετε, το οποίο είναι μοναδικό για όλες τις σελίδες της βάσης. Ο τύπος **PageId** καθορίζετε ως integer στο **minirel.h**.

Όταν μια σελίδα ζητείται από τον διαχειριστή προσωρινής μνήμης τότε συμβαίνουν τα ακόλουθα: Ελέγχετε το buffer pool για να δει κατά πόσο περιέχει ήδη την εν λόγω σελίδα. Εάν η σελίδα δεν περιέχετε στο pool, τότε πρέπει να τη φέρει ακλουθώντας την εξής διαδικασία:

- Διάλεξε ένα frame για αντικατάσταση ακλουθώντας την πολιτική LOVE/HATE.
- Αν το επιλεγμένο προς αντικατάσταση frame είναι dirty τότε γράψε το στο δίσκο (χρησιμοποιώντας την κατάλληλη μέθοδο της κατηγορίας DB).
- Διάβασε την ζητούμενη σελίδα (χρησιμοποιώντας την κατάλληλη μέθοδο της κατηγορίας DB) στο επιλεγμένο προς αντικατάσταση frame. Τα *pin_count* και *dirtybit* για το frame πρέπει να αρχικοποιηθούν στο 0 και FALSE αντίστοιχα.

- Ενημέρωσε την καταχώρηση για αυτό το frame στον πίνακα **bufDescr**.
- Κάνε *Pin* την ζητούμενη σελίδα αυξάνοντας το *pin_count* στο descriptor για αυτό το frame και επέστρεψε ένα δείκτη (pointer) στη σελίδα στον αιτητή.

Παραδοτέα

Πρέπει να παραδώσετε τα **όλα** πηγαία αρχεία σας μέσω του Moodle. Καθυστερημένες υποβολές δεν θα γίνονται αποδεκτές.

Παρακαλώ βεβαιωθείτε ότι το πρόγραμμα σας παράγει το αναμενόμενο αποτέλεσμα συγκρίνοντας τα αποτελέσματα σας με αυτά που βρίσκονται στο αρχείο *correct_output*.

Καλή Επιτυχία!