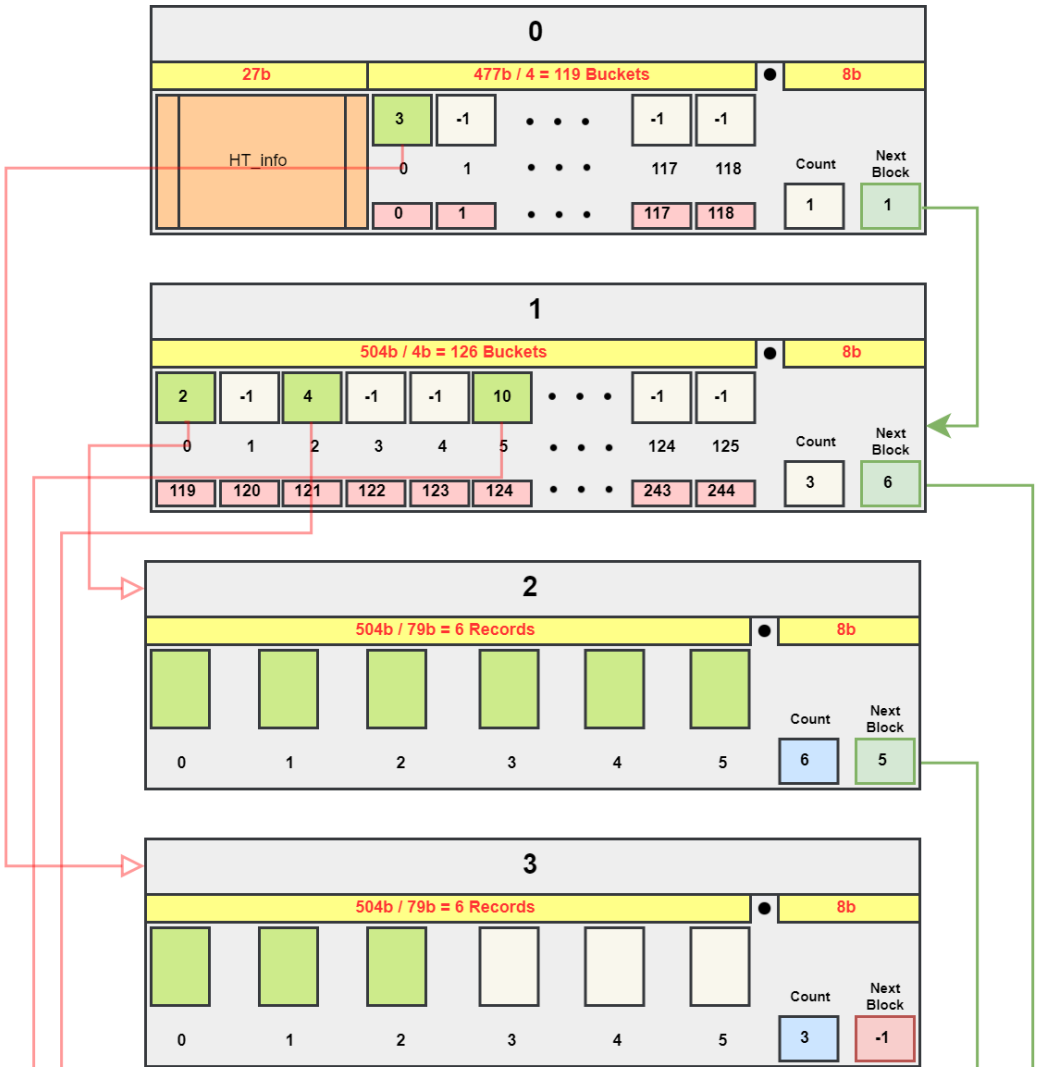
ΕΘΝΙΚΟ ΚΑΙ ΚΑΠΟΔΙΣΤΡΙΑΚΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΘΗΝΩΝ

ΣΧΟΛΗ ΘΕΤΙΚΩΝ ΕΠΙΣΤΗΜΩΝ
ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΤΗΛΕΠΙΚΟΙΝΩΝΙΩΝ

Υλοποίηση Συστημάτων Βάσεων Δεδομένων

Χατζόπουλος Κωνσταντίνος Α.Μ. : 1115201300202	Σκούρας Γεώργιος Α.Μ. : 1115201400309

ΣΤΑΤΙΚΟΣ ΚΑΤΑΚΕΡΜΑΤΙΣΜΟΣ



ΑΘΗΝΑ 2019

Πρωτεύον Ευρετήριο

Στην υλοποίηση του στατικού κατακερματισμού για το πρωτεύον ευρετήριο, υπάρχουν 2 είδη block μεγέθους 512 Bytes. Τα blocks ευρετηρίου και τα blocks εγγραφών. Σε κάθε μπλοκ στο τέλος του υπάρχει χώρος για 2 ακεραίους. Συγκεκριμένα στη θέση `BLOCKSIZE – 2 * sizeof(int)` υπάρχει ο μετρητής (count) και στη θέση `BLOCKSIZE – sizeof(int)` ένας δείκτης για το επόμενο μπλοκ αν έχει.
Σε περίπτωση που δεν υπάρχει επόμενο μπλοκ η τιμή παραμένει 0.

HT_CreateIndex()

Στα blocks ευρετηρίου και συγκεκριμένα στο block 0 αποθηκεύεται η δομή HT_info η οποία έχει πληροφορίες για το ίδιο το ευρετήριο. Ακριβώς μετά, στο ίδιο block τοποθετούνται δείκτες όπου δείχνουν στην αρχή των buckets του ευρετηρίου (hash table).

Κατά τη διάρκεια της αρχικοποίησης του ευρετηρίου, σε περίπτωση που ο αριθμός των απαιτούμενων bucket ξεπερνάει τον μέγιστο χώρο που μπορεί να φιλοξενήσει το εκάστοτε block, τότε ζητήται νέο block από το Block File System και τα υπόλοιπα buckets τοποθετούνται σε αυτό. Αν πάλι δεν χωράνε, η διαδικασία επαναλαμβάνεται με τη βοήθεια της αναδρομικής συνάρτησης `recInitializeBuckets()` και δημιουργούνται όσα blocks χρειάζονται για να χωράει όλο το hash table.

Κάθε block έχει αποθηκευμένο ένα μονοδιάστατο πίνακα από int όπου κάθε θέση του, αντιπροσωπεύει ένα bucket. Συγκεκριμένα, η τιμή του κάθε bucket υποδηλώνει το αρχικό block της αλυσίδας από blocks εγγραφών.

HT_OpenIndex()

Η συνάρτηση αναλαμβάνει να ανοίξει το αρχείο με όνομα filename, διαβάζει το πρώτο block και τέλος αποθηκεύει τις πληροφορίες σε μια δομή HT_info.

HT_CloseIndex(HT_info * info)

Η συνάρτηση αναλαμβάνει να κλείσει το αρχείο με τον αντίστοιχο αναγνωριστικό αριθμό που υπάρχει στη δομή HT_info και τέλος αποδεσμεύει το HT_info από τη μνήμη.

HT_InsertEntry()

Τα blocks εγγραφών έχουν στα τελευταία 2* sizeof(int) bytes τον αριθμό των εγγραφών που υπάρχουν σε αυτό το block και στα επόμενα sizeof(int) bytes τον αριθμό του επόμενου block που συνεχίζονται οι εγγραφές.

Από το HT_info, βλέπουμε ποιο attrType και attrName θα χρησιμοποιήσουμε για να υπολογίσουμε το hash value. Η συνάρτηση αναλαμβάνει να υπολογίσει το hash value της εγγραφής και να υπολογίσει σε ποιο block του hash table πρέπει να μπει αυτή τη εγγραφή.

Στη συνέχεια καλείται η συνάρτηση _insertEntry() η οποία αναλαμβάνει να "γράψει" στο block που πρέπει την εγγραφή. Αν η τιμή του bucket είναι -1 (δηλαδή δεν έχει ξαναμπεί εγγραφή με αυτό το hash value), δεσμεύεται νέο block, γράφεται ο αριθμός του νέου block στο bucket και διαβάζεται το νέο block. Ο αριθμός των εγγραφών ορίζεται σε 1 ενώ ο αριθμός του επόμενου block = -1 και τοποθετείται η εγγραφή στο block αυτό.

Αν η τιμή του κελιού είναι διαφορετική του -1, διαβάζουμε το block με τη τιμή αυτή και ελέγχουμε τον αριθμό των εγγραφών σε αυτό. Άμα είναι μικρότερος του μέγιστου αριθμού εγγραφών σε block, βάζουμε την εγγραφή στο block αυτό και αυξάνουμε τον αριθμό των εγγραφών σε αυτό το block κατά 1.

Διαφορετικά, κοιτάμε εάν υπάρχει άλλο block υπερχείλισης. Μπαίνουμε λοιπόν, σε μια επανάληψη που στοχεύει να βρει το τελευταίο block που υπάρχει για αυτό το hash value. Όταν το βρει, εάν χωράει η εγγραφή θα μπει εκεί και θα αυξήσει τον αριθμό εγγραφών του block, αλλιώς θα δημιουργήσει νέο block να βάλει την εγγραφή και θα φροντίσει να ενημερώσει το προηγούμενο block με τον αριθμό του νέου block και όταν τελειώσει η εκτέλεσή της επιστρέφει το block στο οποίο "γράφτηκε" η εγγραφή.

HT_DeleteEntry()

Από το HT_info, βλέπουμε ποιο attrType και attrName θα χρησιμοποιήσουμε για να υπολογίσουμε το hash value. Η συνάρτηση υπολογίζει το hash value της εγγραφής και υπολογίζει σε ποιο block και σε ποιο bucket του hash table θα πρέπει να ψάξει για να βρει την εκάστοτε εγγραφή.

Στη συνέχεια καλούμε την συνάρτηση _deleteEntry() η οποία αναλαμβάνει να διαγράψει μια εγγραφή με τον εξής τρόπο: όταν βρίσκει την εγγραφή προς διαγραφή, αναζητά την τελευταία εγγραφή από τα block υπερχείλισης και παίρνει τα στοιχεία της τελευταίας εγγραφής και τα βάζει στη θέση της εγγραφής που διαγράφηκε. Έπειτα ο counter του τελευταίου block της αλυσίδας μειώνεται κατά 1.

Κατά τη διαδικασία της διαγραφής δεν διαγράφεται πραγματικά κάποια εγγραφή αλλά ενημερώνονται κατάλληλα οι δείκτες και οι μετρητές, επιπλέον δεν διαγράφονται από τη μνήμη τα άδεια blocks και οι αλυσίδες (blocks υπερχείλισης) παραμένουν ως έχουν.

HT_GetAllEntries()

Από το HT_info, βλέπουμε ποιο attrType και attrName θα χρησιμοποιήσουμε για να υπολογίσουμε το hash value. Η συνάρτηση υπολογίζει το hash value της εγγραφής και υπολογίζει σε ποιο block και σε ποιο bucket του hash table υπάρχει αυτή τη εγγραφή.

Στην συνέχεια καλούμε την συνάρτηση _getAllEntries η οποία αναλαμβάνει να διαβάσει το κατάλληλο block και το κατάλληλο bucket αυτού του block. Αν η τιμή του bucket είναι -1, δεν υπάρχει καμία εγγραφή με αυτό το value. Διαφορετικά, διαβάζει το block με αριθμό της τιμής του bucket, διαβάζει όσες εγγραφές έχει και εκτυπώνει όσες έχουν το value που ψάχνουμε.

Εάν ο αριθμός των εγγραφών είναι ο μέγιστος, ελέχγεται εάν υπάρχει block υπερχείλισης. Εάν υπάρχει, διαβάζονται οι εγγραφές εκείνου του block και εάν δε βρεθεί η εγγραφή που αναζητάται, ελέγχεται εάν υπάρχει και άλλο block για αυτό το hash value. Η επανάληψη αυτή συνεχίζεται εώς ότου βρεθεί block που δεν έχει επόμενο.

Βοηθητικές Συναρτήσεις

HT_PrintIndex()

Η συνάρτηση αυτή έχει υλοποιηθεί για την οπτικοποίηση του αποτελέσματος των ενεργειών που γίνονται στο ευρετήριο.

void _printRecord()

Αναλαμβάνει να εκτυπώσει μια εγγραφή.

void _getCount()

Δίνει τη πληροφορία για το πόσες εγγραφές έχει ένα block.

void _setCount()

Ενημερώνει τον counter των εγγραφών του block σε περίπτωση που υπάρξει μεταβολή στον αριθμό των εγγραφών.

void _getNext()

Δίνει τη πληροφορία για το ποιο είναι το επόμενο block της αλυσίδας (είτε σε επίπεδο εγγραφών είτε σε επίπεδο HashTable).

void _setNext()
Ορίζει το ποιο πρέπει να είναι το επόμενο block της αλυσίδας(είτε σε επίπεδο εγγραφών είτε σε επίπεδο HashTable).

void _getRecord()
Δίνει τη πληροφορία για το ποια εγγραφή βρίσκεται σε συγκεκριμένο bucket ενός block.

void _setRecord()
Αναλαμβάνει να “γράψει” μια εγγραφή στο κατάλληλο bucket ενός block εγγραφών.

int _getMaxRecords()
Δίνει τη πληροφορία για το πόσες εγγραφές “χωράνε” σε ένα block εγγραφών

int _getMaxBuckets()
Δίνει τη πληροφορία για το πόσες buckets “χωράνε” σε ένα block που αφορά το HashTable.

long int _int_h() & long int _char_h()
Είναι συναρτήσεις κατακερματισμού που βρέθηκαν έπειτα από αναζήτηση στο site: http://www.cse.yorku.ca/~oz/hash.html.