ΥΣΒΔ Εργασία

Η εργασία εκπονήθηκε από τους:

Δημήτριος Μυλωνόπουλος – 1115201600112 και
Εμμανουήλ Λύκος – 1115201600096

Οδηγίες:

Για να μεταγλωτίσσετε το πρόγραμμα γράφεται στο terminal την εντολή **make.** Αν έχετε θέλετε να τρέξεται την main του ht_main_test.c τότε για να τρέξετε το πρόγραμμα γράφετε την εντολή **./example_main <number-of-records> <deletion-density (should be between 0 and 1)>** αλλιώς γράφετε την εντολή **./my_main** ή **./ht_main_names** . Για να διαγράψετε τα εκτελέσιμα και τα αντικειμενικά αρχεία γράφετε την εντολή **make clean.**

Σημείωση: Εάν επιθυμείτε να μεταγλωττίσετε κάποιο άλλο main αρχείο αλλάζεται στο makefile το όνομα που αντιστοιχεί στο main αρχείο.

Για το Primary Hash Table:

HT_CreateIndex: Στη Συνάρτηση αυτή αρχικοποείται το κύριο αρχείο κατακερματισμού. Στο 1ο block (block 0) αποθηκεύονται οι πληροφορίες σχετικά με το αρχείο κατακερματισμού ενώ στα blocks (1-αριθμός buckets) αρχικοποιονται τα buckets του hashTable, με υπολοιπόμενο αριθμό εγγραφών
6 που είναι και ο μέγιστος αριθμός εγγραφών που χωράει η δομή στο HashTableBlock.h. Επιπλέον το next_block τους αρχικοποείται με -1 για να δείξουμε πως δεν υπάρχει επόμενο block.

HT_OpenIndex: Σε αυτή την συνάρτηση διαβάζουμε από το Block 0 τις πληροφορίες για το Primary Hash Table τις οποίες αντιγράφουμε σε μια δυναμικά δεσμευμένη δομή HT_info.

HT_CloseIndex: Σε αυτή την συνάρτηση κλείνουμε το ανοιχτό αρχείο κατακερματισμού και ελευθερώνουμε την δυναμικά δεσμευμένη δομή HT_info.

HT_InsertEntry: Σε αυτή τη συνάρτηση εισάγουμε ένα Record στο hash Table. Το κλειδί που θα χρησιμοποιηθεί στην συνάρτηση εξαρτάται απ' το AttrName και το AttrType της δομής του hash table. Αν το AttrType = i τότε ως κλειδί θα χρησιμοποιηθεί το id. Αλλιώς αν AttrType = c τότε ανάλογα με το AttrName αποφασίζουμε αν το κλειδί θα είναι το όνομα, το επώνυμο ή η διεύθυνση της εγγραφής. Η συνάρτηση κάνει hash στο κλειδί και προσθέτει + 1 γιατί η Hash επιστρέφει τιμές απο 0 ενώ τα buckets ξεκινούν από 1. Η συνάρτηση ψάχνει να βρει την πρώτη διαθέσιμη θέση στο Hash Table για αυτό και κάνει itterate στα blocks που συνδέονται με το bucket που επέστρεψε η hash function. Αν δε βρεί κάποιο μη γεμάτο block δημιουργεί ένα καινούργιο block και συνδέει τον "πατέρα" του με αυτό. Η συνάρτηση επιστρέφει τον αριθμό του block στο οποίο έγινε η εισαγωγή της εγγραφής.

HT_DeleteEntry: Η συνάρτηση αυτή ψάχνει μέσα στο Primary Hash Table για την εγγραφή που περιέχει το κλειδί που έχει δωθεί ως όρισμα στη συνάρτηση ώστε να τη διαγράψει. Προφανώς όπως και στην insert ο τύπος του κλειδιού θα εξαρτηθεί από τα AttrName και AttrType. Για την αναζήτηση βρίσκει μέσω της hashFunction το bucket στο οποίο θα έπρεπε να είναι η εγγραφή και κάνει iterrate τα blocks που του αντιστοιχούν. Αν βρεθεί η εγγραφή σε κάποιο απ' αυτά τότε βάζουμε στη θέση της την τελευταία εγγραφή του block και αυξάνουμε τον αριθμό των blocks που απομένουν. Σε περίπτωση που δεν τη βρει η συνάρτηση επιστρέφει -1.

HT_GetAllEntries: Η συνάρτηση αυτή ψάχνει μέσα στο Primary Hash Table για την εγγραφή που περιέχει το κλειδί που έχει δωθεί ως όρισμα στη συνάρτηση. Προφανώς όπως και στην insert ο τύπος του κλειδιού θα εξαρτηθεί από τα AttrName και AttrType. Για την αναζήτηση βρίσκει μέσω της hashFunction το bucket στο οποίο θα έπρεπε να είναι η εγγραφή και κάνει iterrate τα blocks που του αντιστοιχούν. Αν βρεθεί η εγγραφή σε κάποιο απ αυτά τότε την εκτυπώνουμε και επιστρέφουμε τον αριθμό των blocks που χρειάστηκε να ψάξουμε μέχρι να τη βρούμε. Σε περίπτωση που δεν τη βρει η συνάρτηση εκτυπώνει πως η εγγραφή δεν βρέθηκε και επιστρέφει -1.

Σημείωση: Σε όλες τις συναρτήσεις του ht θεωρούμε πως το primary key είναι μοναδικό στα αρχεία προς εισαγωγή, με βάση τον ορισμό του κλειδιού. Γι' αυτό η HT_GetAllEntries και η HT_Delete αναζητούν μόνο μέχρι την πρώτη εγγραφή που θα βρούν.

Για το **Secondary Hash Table** :

SHT_CreateSecondaryIndex: Σε αυτή την συνάρτηση ουσιαστικά αρχικοποιείται το αρχείο δευτερεύοντος κατακερματισμού. Αρχικά, ελέγχεται αν το αρχείο πρωτεύοντος κατακερμάτισμού είναι όντως αρχείο πρωτεύοντος κατακερματισμού,έχει ως κλειδί το ID της εγγραφής και αν το attrName και το attrLength ,για το αρχείο δευτερεύοντος κατακερματισμού, είναι ορθά. Έπειτα, φτιάχνει το πρώτο block έτσι ώστε να περιέχει την απαραίτητη(και ζητούμενη) πληροφορία για το αρχείο και φτιάχνει τόσα blocks όσα είναι τα buckets του secondary hash table για να ξέρουμε από ποιο block ξεκινάει κάθε bucket. Τέλος, διαβάζει όλα τα block του πρωτεύοντος αρχείου κατακερματισμού και για κάθε έγγραφη που βρίσκει την προσθέτει στο αρχείο δευτερεύοντος κατακερματισμού γιατί απλά το δευτερεύον αρχείο κατακερματισμού λειτουργεί ως ευρετήριο για το πρωτεύον.

SHT_OpenSecondaryIndex: Σε αυτή την συνάρτηση ουσιαστικά ανοίγει το αρχείο δευτερεύοντος κατακερματισμού, διαβάζει την πληροφορία του πρώτου block, την αντιγράφει στην δυναμική δομή SHT_info και επιστρέφει δείκτη σε αυτή την δομή.

SHT_CloseSecondaryIndex: Αυτή η συνάρτηση απλά κλείνει το αρχείο δευτερεύοντος κατακερματισμού και αποδεσμεύει την μνήμη που δεσμεύτηκε από την δομή SHT_info στην SHT_OpenSecondaryIndex.

SHT_SecondaryInsertEntry: Σε αυτή την συνάρτηση αρχικά βλέπω το attrName για να ξέρω ποιο μέλος της εγγραφής είναι το κλειδί και εφαρμόζοντας την συνάρτηση κατακερματισμού βρίσκω σε ποιο bucket πρέπει να μπει η εγγραφή. Έπειτα, μετασχηματίζω την δομή SecondaryRecord σε RealSecondaryRecord καθώς αυτή θα μπει στο πρώτο διαθέσιμο block ενός bucket καθώς για να βρω την εγγραφή στο πρωτεύον hash table απλά χρειάζομαι τον αριθμό του block του primary hash table στο οποίο μπήκε η εγγραφή για να την βρω πιο γρήγορα, την συμβολοσειρά που αποτελεί το κλειδί της δευτερεύουσας εγγραφής για να μπορώ να την βρω στο δευτερεύον hash table και το ID της εγγραφής καθώς αυτό προσδίδει μοναδικότητα σε κάθε εγγραφή επομένως θα είναι ένα σωστό κριτήριο αναζήτησης στο πρωτεύον hash table. Η διαδικασία εισαγωγής είναι ίδια με την διαδικασία εισαγωγής σε απλά συνδεδεμένη λίστα στην οποία κάθε κόμβος χωράει 10 εγγραφές.

Σημείωση: Σε κάθε block χωράνε 10 εγγραφές διότι το μέγεθος της δομής RealSecondaryRecord είναι 48 bytes και οι άλλοι 3 ακέραιοι καταλαμβάνουν 12 bytes μαζί. Άρα η συνολική δομή του block καταλαμβάνει χώρο 480 + 12 = 492 bytes.

SHT_SecondaryGetAllEntries: Η παρούσα συνάρτηση απλά για το bucket που αντιστοιχεί στο κλειδί value διατρέχει όλα τα blocks που αντιστοιχούν σε αυτό και για κάθε εγγραφή που ανήκει στο παρών block και έχει τιμή στο κλειδί της ίση με value, διαβάζει το block του

πρωτεύοντος αρχείου κατακερματισμού που υποτίθεται πως υπάρχει η εγγραφή και αν όντως υπάρχει τότε την εκτυπώνει. Για να δουλέψει η παρούσα συνάρτηση προϋποθέτει ότι το αρχείο πρωτεύοντος κατακερματισμού είναι ανοικτό.

Για τις **main:**

ht_main_test.c: Είναι η ενδεικτική main που μας δίνετε.

main.c : Για να δουλέψει το αρχείο πρέπει να έχει τουλάχιστον όσες εγγραφές είναι defined στο πρόγραμμα γιατί επεξεργάζεται τόσες εγγραφές. Επίσης, το hash table και το secondary hash table έχει τόσα buckets όσα έχουν γίνει defined στο πρόγραμμα. Η παρούσα main αρχικά προσθέτει τις πρώτες μισές εγγραφές μόνο στο πρωτεύον hash table ενώ τις άλλες μισές εγγραφές και στο δευτερεύον ευρετήριο. Έπειτα, τις αναζητά όλες στο πρωτεύον hash table και αν δεν βρει κάποια τότε μάλλον κάτι δεν πήγε καλά και σταματάει η εκτέλεση του προγράμματος. Έπειτα, για κάθε όνομα που εισήχθη στο δευτερεύον ευρετήριο κάνουμε αναζήτηση και προφανώς αν δεν βρεθεί κάποιο τότε κάτι δεν πήγε καλά. Επίσης, διαγράφουμε όσες τυχαίες εγγραφές έχουμε κάνει define στο πρόγραμμα και ξανακάνουμε αναζήτηση στο πρωτεύον hash table και αν δεν βρεθούν όσες εγγραφές έχουμε διαγράψει όλα πήγαν καλά.

ht_main_names_test.c: Η παρούσα main συνάρτηση δουλεύει όπως η **./main.c** αλλά το μόνο που κάνει είναι να φτιάχει ένα Primary Hash Table με πεδίο-κλειδί το όνομα της εγγραφής και ουσιαστικά στην αρχή βάζει RECORDS εγγραφές στο hash table, τις αναζητά, διαγράφει μερικές και πάλι τις αναζητά και περιμένουμε να μην βρει όσες έχουμε διαγράψει.

HashStatistics: Η συνάρτηση αυτή επιστρέφει τα στατιστικά στοιχεία της εκτέλεσης των Primary και Secondary hash tables. Για να αναγνωρίσουμε το είδος του αρχείου στο block 0 και των δύο συναρτήσεων κρατούμε μια μεταβλητή η οποία αρχικοποιείται με 'p' ή 's' ανάλογα με το αν είναι primary ή secondary αντίστοιχα. Σε περίπτωση που δεν είναι αρχείο κατακερματισμού επιστρέφει -1. Η συνάρτηση αυτή κάνει itterate στα buckets του hash table και αντίστοιχα στα blocks που αντιστοιχούν στο κάθε bucket ώστε να συλλέξει τα στατιστικά. Εκτυπώνει τον μέγιστο τον ελάχιστο και τον μέσο αριθμό εγγραφών ανά bucket. Τον μέσο αριθμό blocks ανα bucket, το οποίο είναι τα συνολικά blocks -1 για το block 0 πρός τον αριθμό των buckets. Τέλος εκτυπώνει τον αριθμό των buckets που έχουν υπερχείληση αλλά και τον αριθμό των overflow blocks για το κάθε bucket.