Υλοποίηση Συστημάτων Βάσεων Δεδομένων
Εργασία 2

Μέλη ομάδας:

- Ιωάννης Καπετανγεώργης
  Αριθμός Μητρώου: 1115201800061

- Δημήτριος Σιταράς
  Αριθμός Μητρώου: 1115201800178

Εξάμηνο: 5ο

Δομή Αρχείων

Στο παραδοτέο directory περιέχονται:

- Το directory src το οποίο περιέχει τα αρχεία HT.c και SHT.c με τις υλοποιήσεις του πρωτεύοντος ευτηρίου στατικού κατακερματισμού και δευτερεύοντος ευτηρίου στατικού κατακερματισμού αντίστοιχα.
- Το directory include το οποίο περιέχει τα αρχεία BF.h, HT.h και SHT.h τα οποία γίνονται include από τα αντίστοιχα .c αρχεία του φακέλου src.
- Το directory examples το οποίο περιέχει την main συνάρτηση η οποία επιδεικνύει την λειτουργία του πρωτεύοντος και του δευτερεύοντος ευρετηρίου στατικού κατακερματισμού. Η ακριβής λειτουργία του αρχείου αυτού εξηγείται αναλυτικότερα παρακάτω.
- Το directory build το οποίο περιέχει το εκτελέσιμο που δημιουργείται έπειτα από την μεταγλώττιση. Πιο συγκεκριμένα, δημιουργείται το εκτελέσιμο demoSHT.
- Το directory lib το οποίο περιέχει τα αρχεία BF_64.a και BF_32.a που μας δόθηκαν και περιέχουν την υλοποίηση της βιβλιοθήκης BF.
- Το αρχείο Makefile μέσω του οποίου γίνεται το compilation των αρχείων. Το περιεχόμενο του Makefile εξηγείται στην επόμενη παράγραφο.

Μεταγλώττιση του προγράμματος

Μέσα στο αρχικό directory πληκτρολογούμε στο τερματικό την εντολή make (αν θελουμε να διαγραψουμε τα αρχεία που παράγονται απλά πληκτρολογούμε make clean). Έτσι, δημιουργείται το εκτελέσιμο demoSHT στο directory build.

Εκτέλεση του προγράμματος

Πηγαίνουμε στο directory /build και “τρέχουμε” το εκτελέσιμο demoSHT.

Υλοποίηση Πρωτεύοντος Ευρετηρίου Στατικού Κατακερματισμού(./src/HT.c)

Δεν έχουμε αλλάξει τίποτα στην υλοποίηση του πρωτεύοντος ευτηρίου στατικού κατακερματισμού (HT.c) σε σχέση με την πρώτη εργασία. Η υλοποίηση εξηγείται αναλυτικά στο Readme της πρώτης εργασίας το οποίο συμπεριλαμβάνεται επίσης στα παραδοτέα αρχεία.

Υλοποίηση Δευτερεύοντος Ευρετηρίου Στατικού Κατακερματισμού(./src/SHT.c)

SHT_CreateSecondaryIndex(...)

Η παραπάνω συνάρτηση δημιουργει και αρχικοποιεί το δευτερεύον ευρετήριο στατικού κατακερματισμού με το αντίστοιχο όνομα που δίνεται ως όρισμα.
Αρχικά, δημιουργείται το αρχείο μέσω της BF_OpenFile. Στη συνέχεια, δεσμεύεται ένα block στο οποιο αποθηκεύουμε την επικεφαλίδα του αρχείου κατακερματισμού (SHT_info) καθώς και ένας pointer στο επόμενο block το οποίο περιέχει τον πίνακα κατακερματισμού (ή μέρος αυτού) και αποθηκεύονται οι αλλαγές στο block αυτό μέσω της BF_WriteBlock.
Στην συνέχεια δημιουργούμε και αρχικοποιούμε τον πίνακα κατακερματισμού. Ο πίνακας ενδεχομένως να χρειάζεται παραπάνω από ένα block για να αποθηκευτεί. Έτσι, μέσω μιας επαναληπτικής διαδικασίας κάθε φορά δημιουργούμε ένα block και αποθηκεύουμε τον μέγιστο αριθμό blocks που χωράνε στο block αυτό. Η διαδικασία αυτή ολοκληρώνεται όταν έχουμε δημιουργήσει τον αριθμό των buckets ο οποίος δόθηκε σαν όρισμα. Προφανώς αν ο πίνακας κατακερματισμού χωράει ολόκληρος σε ένα block, απλά δημιουργούμε ένα block στο οποίο και αποθηκεύουμε τα buckets που θέλουμε.
Κάθε ένα από αυτά τα blocks αρχικά περιέχει έναν ακέραιο ο οποίος μας “δείχνει” το πόσα buckets περιέχει το block αυτό. Επίσης περιέχει έναν pointer στο επόμενο block το οποίο περιέχει τον υπόλοιπο πίνακα κατακερματισμού. Σε περίπτωση που ο πίνακας κατακερματισμού “χωράει” σε μόνο ένα block, αλλά και στην περίπτωση του block που περιέχει το τελευταίο κομμάτι του πίνακα, ο pointer αυτός περιέχει την τιμή -1. Έπειτα από τους αριθμούς αυτούς αποθηκεύονται τα buckets. Το κάθε bucket ουσιαστικά είναι ένας ακέραιος ο οποίος μας “δείχνει” στο block το οποίο περιέχει τις εγγραφές που αντιστοιχούν στο bucket αυτό. Το κάθε bucket αρχικοποιείται με την τιμή -1 καθώς στην τρέχουσα κατάσταση δεν υπάρχει κανένα block με εγγραφές.
Σε περίπτωση που τα παραπάνω εκτελεστούν επιτυχώς, η συνάρτηση επιστρέφει 0. Αντίθετα, σε περίπτωση λάθους η συνάρτηση επιστρέφει -1.

SHT_OpenSecondaryIndex(...)

Η παραπάνω συνάρτηση απλά “ανοίγει” το ήδη δημιουργηθέν δευτερεύον ευρετήριο στατικού κατακερματισμού με το όνομα που δόθηκε ως όρισμα και διαβάζει το πρώτο block το οποίο περιέχει την επικεφαλίδα του δευτερεύον ευρετήριο στατικού κατακερματισμού. Στην συνέχεια δημιουργεί ένα αντίγραφο της επικεφαλίδας αυτής το οποίο και επιστρέφει. Σε περίπτωση λάθους η συνάρτηση επιστρέφει NULL.

SHT_CloseSecondaryIndex(...)

Η παραπάνω συνάρτηση απλά “κλείνει” το ήδη δημιουργηθέν δευτερεύον ευρετήριο στατικού κατακερματισμού με το όνομα που δόθηκε ως όρισμα. Επίσης, αποδεσμεύει το αντίγραφο της επικεφαλίδας το οποίο είχε δημιουργηθεί (και δεσμευθεί δυναμικά) από την SHT_OpenSecondaryIndex. Σε περίπτωση που τα παραπάνω εκτελεστούν επιτυχώς, η συνάρτηση επιστρέφει 0. Αντίθετα, σε περίπτωση λάθους η συνάρτηση επιστρέφει -1.

SHT_SecondaryInsertEntry(...)

Η παραπάνω συνάρτηση εισάγει στον δευτερεύον πίνακα κατακερματισμού μια νέα εγγραφή που δίνεται ως όρισμα. Η συνάρτηση σύμφωνα με την εκφώνηση δέχεται σαν όρισμα ένα αντικείμενο τύπου SecondaryRecord. Στο δευτερεύον ευρετήριο χρειάζεται να αποθηκεύουμε μόνο το surname και το αντιστοιχο block (το οποίο εισήχθει το record στο πρωτευον ευρετηριο) για κάθε record. Επομένως τελικά στο δευτερεύον ευρετήριο αποθηκεύουμε αντικείμενα τύπου secRec (η δήλωση υπάρχει στο αρχείο SHT.h) έτσι ώστε να αποθηκεύσουμε μόνο τις δύο αυτές πληροφορίες και όχι επιπλέον περιττές τις οποίες περιέχει το SecondaryRecord. Βέβαια, πρέπει να έχει προηγηθεί η εισαγωγή του κάθε record στο πρωτεύον ευρετήριο και έπειτα να γίνεται η αντίστοιχη εισαγωγή του στο δευτερεύον ευρετήριο.
Κάθε block εγγραφών περιέχει στην αρχή έναν ακέραιο ο οποίος “κρατάει” τον αριθμό τον εγγραφών που είναι αποθηκευμένες την τρέχουσα στιγμή στο συγκεκριμένο block. Επίσης, μετά από τον ακέραιο αυτό, αποθηκεύεται ακόμα ένας ακέραιος ο οποίος λειτουργεί σαν pointer στο επόμενο block έτσι ώστε να δημιουργηθεί μια “αλυσίδα” για τα overflow blocks. Στη συνέχεια, μέσω της συνάρτησης κατακερματισμου hash(...) (περισσότερα στις παρατηρήσεις), στην οποία δίνουμε ως όρισμα το κλειδί της εγγραφής που θέλουμε να εισάγουμε, βρίσκουμε το αντίστοιχο bucket του πινακα που πρέπει να εισαχθεί η εγγραφή. Έπειτα, αναζητουμε το block στο οποίο βρίσκεται το bucket αυτό (καθώς ο πίνακας κατακερματισμού μπορεί να είναι αποθηκευμένος σε πολλαπλά blocks). Σε περίπτωση που το bucket αυτό δεν δείχνει σε κανένα block, δηλαδή δεν περιέχει καμία εγγραφή, αρχικά χρειάζεται να δημιουργήσουμε ένα νέο block. Στο block αυτό αρχικοποιούμε τον μετρητή εγγραφών σε 1, τον pointer για το επόμενο block σε -1 (καθώς δεν υπάρχει overflow block) και τέλος αποθηκεύουμε το record έπειτα από τους δύο αυτούς ακέραιους. Τέλος, ανανεώνουμε την τιμή του bucket με τον αριθμό του block που μόλις δημιουργήσαμε, έτσι ώστε πλέον το bucket να “δείχνει” στο block αυτό. Ειδάλλως, το bucket δείχνει σε ένα block εγγραφών, το οποίο ίσως και να “δείχνει” σε μια αλυσίδα από overflow blocks. Έτσι, ξεκινάμε να ανατρέχουμε ένα προς ένα τα blocks της “αλυσίδας”. Σε κάθε ένα block ελέγχουμε αν υπάρχει διαθέσιμος χώρος για να αποθηκεύσουμε την νέα εγγραφή. Σε περίπτωση που υπάρχει χώρος σε ένα ήδη υπάρχον block, αποθηκεύουμε την νέα εγγραφή μετά από τις ήδη υπάρχουσες, αυξάνουμε τον μετρητή εγγραφών που υπάρχει στην αρχή του block και τέλος επιστρέφουμε τον αριθμό του block στο οποίο μόλις αποθηκεύσαμε το νέο record. Σε περίπτωση που φτάσουμε στο τέλος της αλυσίδας και δεν έχουμε βρει διαθέσιμο χώρο για να εισάγουμε την εγγραφή, χρειάζεται να δημιουργήσουμε ένα νέο block. Συνεπώς, δεσμεύουμε το νέο block, αρχικοποιούμε τον μετρητή εγγραφών σε 1, αρχικοποιούμε τον pointer του επόμενου block σε -1 (καθώς είναι το τελευταίο block της αλυσίδας) και τέλος αποθηκεύουμε το record έπειτα από τους δύο ακέραιους. Επίσης στον pointer του πρώην

τελευταίου block αποθηκεύουμε τον αριθμό του νέου block που δημιουργήσαμε έτσι ώστε να “συνδεθεί” στην αλυσίδα το νέο τελευταίο block. Έπειτα από όλα αυτά επιστρέφουμε τον αριθμό του νέου block που δημιουργήσαμε και στο οποίο αποθηκεύεται η νέα εγγραφή.

Σημείωση 1η:
Όπως προτάθηκε στο μάθημα σε περίπτωση που υπάρχουν δύο εγγραφές με ίδιο surname και έχουν εισαχθεί στο ίδιο block του πρωτεύοντος ευρετηρίου, πρέπει οι δύο αυτές εγγραφές να αντιστοιχούν σε μία μόνο εγγραφή στο δευτερεύον ευρετήριο.
Έτσι, κάθε φορά που εισάγουμε στο δευτερεύον ευρετήριο ελέγχουμε εάν υπάρχει εγγραφή με ίδιο surname και ίδιο block. Στην περίπτωση αυτή, δεν εισάγεται δεύτερη φορά.
Προφανώς, στο δευτερεύον ευρετήριο μπορεί να υπάρχουν δύο εγγραφές με ίδιο surname αλλά διαφορετικό block.

Σημείωση 2η:
Κάθε φορά που εισάγεται ένα secRec στο ευρετήριο εκτυπώνεται ένα αντίστοιχο μήνυμα.
Επίσης, σε περίπτωση που υπάρχει ήδη ένα secRec με ίδιο surname και ίδιο block, εκτυπώνεται ένα αντίστοιχο ενημερωτικό μήνυμα.

SHT_SecondaryGetAllEntries(...)

Η παραπάνω συνάρτηση αναζητά στον δευτερεύον πίνακα κατακερματισμού όλες τις εγγραφές που έχουν το ίδιο κλειδί (surname) με αυτό που δόθηκε ως όρισμα.
Αρχικά, καλούμε την συνάρτηση hash έτσι ώστε να βρούμε σε ποιο bucket του πίνακα κατακερματισμού βρίσκεται η εγγραφή που ψάχνουμε.
Στην συνέχεια ανατρέχουμε τα blocks στα οποία είναι αποθηκευμένο το hash table έτσι ώστε να βρούμε το bucket αυτό.
Μόλις βρούμε το bucket ελέγχουμε αν “δείχνει” σε κάποιο κάποιο block εγγραφών. Σε περίπτωση που δεν δείχνει σε κάποιο block, αυτό σημαίνει ότι δεν υπάρχει καμία εγγραφή που να αντιστοιχεί στο bucket αυτό, δηλαδή δεν υπάρχει και η εγγραφή που ψάχνουμε. Έτσι, επιστρέφουμε -1.
Σε περίπτωση που το bucket δείχνει σε ένα block, τότε ανατρέχουμε στο block αυτό. Ελέγχουμε μια προς μία τις εγγραφές αυτού του block συγκρίνοντας το κλειδί της εγγραφής που ψάχνουμε με το κλειδί της εγγραφής που ελέγχουμε κάθε φορά. Αν βρούμε την εγγραφή αυτή τότε μέσω του blockId το οποίο αντιστοιχεί στην εγγραφή αυτή, ανατρέχουμε στο πρωτεύον ευρετήριο για να “διαβάσουμε” την εγγραφή αυτή. Πρακτικά, απλά διαβάζουμε το block αυτό (το οποίο αντιστοιχεί στο block στο οποίο είναι αποθηκευμένη στο πρωτεύον ευρετήριο η εγγραφή που ψάχνουμε) μέσω της BF_ReadBlock και βρίσκουμε την εγγραφή του πρωτεύοντος ευρετηρίου. Αν υπάρχουν πολλαπλές εγγραφές με το ίδιο surname στο block αυτό, τότε τυπώνονται όλες.
Σε περίπτωση που δεν βρούμε την εγγραφή σε αυτό το block ακολουθούμε την ίδια ακριβώς διαδικασία επαναληπτικά για τα overflow blocks (εάν αυτά υπάρχουν). Τα blocks αυτά τα προσπελαύνουμε μέσω του "pointer" που περιέχεται σε κάθε block και “δείχνει” στο επόμενο block της αλυσίδας.
Αν ανατρέξουμε όλη την αλυσίδα χωρίς να βρούμε την εγγραφή που ψάχνουμε, σημαίνει ότι η εγγραφή δεν υπάρχει στο αρχείο κατακερματισμού. Έτσι, επιστρέφουμε -1.

HashStatistics(...)

Η συνάρτηση αυτή δέχεται ως όρισμα το όνομα ενός αρχείο κατακερματισμού. Εντοπίζει αν το όνομα αυτό αντιστοιχεί σε αρχείο πρωτεύοντος ή δευτερεύοντος ευρετηρίου και καλεί την HT_HashStatistics ή την SHT_HashStatistics αντίστοιχα.
Η HT_HashStatistics είναι η ίδια με την αυτήν της πρώτης εργασίας απλά με νέο όνομα, η οποία εξηγείται στο αντίστοιχο ReadMe. Ομοίως και η SHT_HashStatistics με μονη διαφορα την προσπέλαση των blocks καθώς περιέχονται διαφορετικού μεγέθους εγγραφές στο δευτερεύον ευρετήριο.
Αρκεί, λοιπόν, να δοθεί απλά ως όρισμα το αρχείου κατακερματισμού του οποίου επιθυμούμε να εκτυπώσουμε τα στατιστικα στοιχεία.

Παρατηρήσεις:

- Η συνάρτηση κατακερματισμού που χρησιμοποιούμε είναι μια παραλλαγή της γνωστής συνάρτησης κατακερματισμού *djb2.*

- Η δομές των records τα οποία εισάγονται στα αρχεία κατακερματισμού, δηλώνονται στα αντίστοιχα αρχεία επικεφαλίδας HT.h και SHT.h και είναι ίδια με αυτά που περιγράφονται στην εκφώνηση της άσκησης.

- Η δομή η οποία περιέχει την επικεφαλίδα του αρχείο κατακερματισμού (SHT_info) επίσης δηλώνεται στο αρχείο SHT.h

- Η αρχικοποίηση της βιβλιοθήκης BF γίνεται μια φορά στην αντίστοιχη main που χρησιμοποιεί τις παραπάνω συναρτήσεις (με την κλήση της συνάρτησης BF_Init()) για την δοκιμή των αρχείων κατακερματισμού.

Συνάρτηση Main
(./examples/main_SHT.c)

main_SHT.c

Αφότου αρχικοποιήσουμε την βιβλιοθήκη BF μέσω της συνάρτησης BF_Init(), δημιουργούμε το αρχείο πρωτεύοντος στατικού κατακερματισμού μέσω της HP_CreateFile και το αρχείο δευτερεύοντος στατικού κατακερματισμού μέσω της SHT_CreateSecondaryIndex. Στην συνέχεια, ανοίγουμε το αρχείο ενδεικτικών εγγραφών που μας έχει δοθεί στο οποίο έχουμε προσθέσει και κάποιες επιπλέον εγγραφές με ίδια surnames και εισάγουμε 1000 εγγραφες στο αρχείο πρωτεύοντος στατικού κατακερματισμού και επειτα στο αρχείο δευτερεύοντος στατικού κατακερματισμού που δημιουργήσαμε δίνοντας σαν όρισμα το block το οποίο μας επέστρεψε η HT_InsertEntry. Προφανώς σε περίπτωση που η HT_InsertEntry επιστρέψει -1 (δηλαδή η εισαγωγή δεν ήταν επιτυχής), δεν εισάγουμε κάτι στο δευτερεύον ευρετήριο.
Έπειτα, επιλέγουμε τυχαία 15 records από αυτά που εισάγαμε στα αντίστοιχα ευρετηρια και για κάθε ένα από αυτά κάνουμε:

- Ελέγχουμε αν η εγγραφή αυτή υπάρχει στο αρχείο πρωτεύοντος στατικού κατακερματισμού μέσω της συνάρτησης HP_GetAllEntries() και εκτυπώνουμε το αντίστοιχο μήνυμα.

Στην συνέχεια επιλέγουμε τυχαία 15 surnames από αυτά που εισάγαμε στα αντίστοιχα ευρετηρια και για κάθε ένα από αυτά κάνουμε:

- Ελέγχουμε αν η εγγραφή αυτή υπάρχει στο αρχείο δευτερεύοντος στατικού κατακερματισμού μέσω της συνάρτησης SHT_SecondaryGetAllEntries() και εκτυπώνουμε το αντίστοιχο μήνυμα.

Στην συνέχεια κλείνουμε τα δύο αρχεία κατακερματισμού μέσω των συναρτήσεων HT_CloseFile και SHT_CloseSecondaryIndex.
Τέλος καλούμε την συνάρτηση HashStatistics για κάθε ένα από τα δύο αρχεία κατακερματισμού έτσι ώστε να εκτυπώσουμε τα στατιστικά που ζητούνται.
Έτσι έχουμε ελέγξει την λειτουργικότητα όλων των συναρτήσεων που υλοποιήσαμε.