Πληροφορική & Τηλεπικοινωνίες
Υλοποίηση Συστημάτων Βάσεων Δεδομένων
Χειμερινό Εξάμηνο 2020 – 2021

Γιώργος – Σάββας Δημητρίου 1115201800045
Χριστίνα Διαμαντή 1115201800046

- Άσκηση 1 -

Η συγκεκριμένη εργασία επικεντρώνεται στην εσωτερική λειτουργία των Συστημάτων Βάσεων Δεδομένων, όσον αφορά τη διαχείριση σε επίπεδο μπλοκ (block) αλλά και ως προς τη διαχείριση σε επίπεδο εγγραφών. Πιο συγκεκριμένα, υλοποιήσαμε ένα σύνολο συναρτήσεων που διαχειρίζονται αρχεία που δημιουργούνται και οργανώνονται σε δύο μορφές: αρχεία σωρού (heap files) και αρχεία στατικού κατακερματισμού (hash files).

Heap Files

Αρχικά, αξίζει να αναφερθεί ότι εκτός του πρώτου μπλοκ, κάθε μπλοκ που δημιουργούμε σε ένα heap file αποθηκεύει τον αριθμό των εγγραφών που περιέχει στα πρώτα τέσσερα bytes του. Αναλυτικότερα, όσον αφορά την δημιουργία αρχείων σωρού (heap files) υλοποιήθηκαν οι συναρτήσεις που ζητούνται.

HP_CreateFile: Δημιουργεί και ανοίγει ένα νέο αρχείο σωρού. Δεσμεύει ένα μπλόκ (το πρώτο) στο οποίο, αφού το διαβάσει, αποθηκεύει με την σειρά μόνο τα ακόλουθα δεδομένα.

1. Συμβολοσειρά “Heap”
2. Τύπος πεδίου-κλειδιού
3. Μήκος κλειδιού
4. Όνομα κλειδιού

Έπειτα γράφει το μπλοκ στον δίσκο και κλείνει το αρχείο.

HP_OpenFile: Ανοίγει ένα αρχείο σωρού και διαβάζει το πρώτο μπλοκ του. Στη συνέχεια ανακτά τη συμβολοσειρά που αποθηκεύσαμε κατά την δημιουργία του αρχείου. Αν αυτή είναι διαφορετική της “Heap”, εκτυπώνει μήνυμα λάθους και επιστρέφει NULL. Διαφορετικά δεσμεύει μνήμη στο σωρό, στην οποία αποθηκεύει τις πληροφορίες του κλειδιού, καθώς επίσης και το file descriptor του αρχείου. Όλα αυτά τα δεδομένα είναι αποθηκευμένα σε μια δομή HP_info , η οποία επιστρέφεται. Τέλος,

υπάρχει δυνατότητα εκτύπωσης των παραπάνω πληροφοριών με την βοήθεια της συνάρτησης printFileInfo.

HP_CloseFile: Κλείνει ένα αρχείο σωρού και αποδεσμεύει την μνήμη που δεσμεύτηκε για τις πληροφορίες του αρχείου.

HP_InsertEntry: Αρχικά λαμβάνουμε τον αριθμό των μπλοκς. Στην συνέχεια αν αυτός ισούται με 1, σημαίνει ότι το αρχείο δεν έχει καθόλου εγγραφές. Οπότε δεσμεύουμε ένα μπλοκ το οποίο διαβάζουμε και αποθηκεύουμε στα πρώτα τέσσερα bytes τον αριθμό 1 (αριθμός εγγραφών) και στα επόμενα bytes την ίδια την εγγραφή.

Αν το αρχείο έχει ήδη εγγραφές, πρώτα διαβάζουμε το τελευταίο μπλοκ του αρχείου και ανακτούμε το πλήθος των εγγραφών. Αν σε αυτό το μπλοκ χωράει η νέα εγγραφή, την αποθηκεύουμε. Σε αντίθετη περίπτωση, δεσμεύουμε ένα νέο μπλοκ και την αποθηκεύουμε εκεί. Σε κάθε περίπτωση, ενημερώνουμε κατάλληλα την μεταβλητή που αποθηκεύει το πλήθος των εγγραφών.

HP_DeleteEntry: Αρχικά λαμβάνουμε τον αριθμό των μπλοκς. Έπειτα, διατρέχουμε κάθε εγγραφή σε κάθε μπλοκ, σειριακά, μέχρι να βρούμε την εγγραφή με το επιθυμητό κλειδί για διαγραφή. Αν το επιθυμητό κλειδί είναι NULL, διαγράφουμε την πρώτη εγγραφή.

Η διαγραφή γίνεται με τον εξής τρόπο:
Το ID της εγγραφής γίνεται 0 και τα υπόλοιπα πεδία-συμβολοσειρές γίνονται “NullString” .
Το παραπάνω επιτυγχάνεται μέσω της βοηθητικής συνάρτησης nullify.

HP_GetAllEntries: Αρχικά λαμβάνουμε τον αριθμό των μπλοκς. Έπειτα, διατρέχουμε κάθε εγγραφή σε κάθε μπλοκ, σειριακά. Κάθε εγγραφή που βρίσκουμε με το επιθυμητό κλειδί, την εκτυπώνουμε. Αν το επιθυμητό κλειδί είναι NULL, εκτυπώνουμε όλες τις εγγραφές.

Hash Files

Αρχικά, αξίζει να αναφερθεί ότι εκτός του πρώτου μπλοκ, κάθε μπλοκ που δημιουργούμε σε ένα hash file, αποθηκεύει τον αριθμό των εγγραφών που περιέχει στα πρώτα τέσσερα bytes του και το ID του μπλοκ με το οποίο συνδέεται, στα επόμενα τέσσερα bytes του. Το ID ενός μπλοκ προκύπτει από την σειρά μ την οποία δεσμεύτηκε (για παράδειγμα το πρώτο μπλοκ έχει ID ίσο με 0, το δεύτερο ίσο με 1 κοκ.). Αναλυτικότερα, όσον αφορά την δημιουργία αρχείων κατακερματισμού (hash files) υλοποιήθηκαν οι συναρτήσεις που ζητούνται.

HT_CreateIndex: Δημιουργεί και ανοίγει ένα νέο αρχείο κατακερματισμού. Δεσμεύει ένα μπλόκ (το πρώτο) στο οποίο, αφού το διαβάσει, αποθηκεύει με την σειρά μόνο τα ακόλουθα δεδομένα.

1. Συμβολοσειρά “Hash”
2. Τύπος πεδίου-κλειδιού
3. Μήκος κλειδιού
4. Όνομα κλειδιού
5. Αριθμός κουβάδων

Έπειτα δεσμεύει άλλα τόσα μπλοκς, όσα και ο αριθμός των κουβάδων και αποθηκεύει στο κάθε ένα τον αριθμό 0 στα πρώτα, καθώς και στα δεύτερα τέσσερα bytes. Αυτό υποδηλώνει ότι ο αριθμός των εγγραφών σε κάθε μπλοκ είναι 0, και το μπλοκ με το οποίο κάθε μπλοκ συνδέεται, είναι το μπλοκ 0 (δηλαδή το πρώτο μπλοκ του αρχείου). Από συμβιβασμό, αυτό δείχνει ότι δεν υπάρχει σύνδεση με κάποιο επόμενο μπλοκ. Δηλαδή ένα μπλοκ το οποίο συνδέεται με το μπλοκ 0 είναι το τελευταίο μπλοκ ενός κουβά. Τέλος, όλες οι πληροφορίες για αυτά τα μπλοκς αποθηκεύονται στον δίσκο.

HT_OpenIndex: Ανοίγει ένα αρχείο κατακερματισμού και διαβάζει το πρώτο μπλοκ του. Στη συνέχεια ανακτά τη συμβολοσειρά που αποθηκεύσαμε κατά την δημιουργία του αρχείου. Αν αυτή είναι διαφορετική της “Hash”, εκτυπώνει μήνυμα λάθους και επιστρέφει NULL. Διαφορετικά δεσμεύει μνήμη στο σωρό, στην οποία αποθηκεύει τις πληροφορίες του κλειδιού, καθώς επίσης, το file descriptor του αρχείου και τον αριθμό των κουβάδων. Όλα αυτά τα δεδομένα είναι αποθηκευμένα σε μια δομή HT_info , η οποία επιστρέφεται. Τέλος, υπάρχει δυνατότητα εκτύπωσης των παραπάνω πληροφοριών με την βοήθεια της συνάρτησης printFileInfo.

HT_CloseIndex: Κλείνει ένα αρχείο κατακερματισμού και αποδεσμεύει την μνήμη που δεσμεύτηκε για τις πληροφορίες του αρχείου.

HT_InsertEntry: Αρχικά κατακερματίζουμε το ID της εγγραφής. Αυτό γίνεται μέσω της συνάρτησης hashIntegers. Διαβάζουμε το μπλοκ με ID το αποτέλεσμα του κατακερματισμού. Μετά ανακτούμε το ID του μπλοκ σύνδεσης. Όσο αυτό είναι διάφορο του 0, διαβάζουμε το μπλοκ σύνδεσης και κάνουμε το ίδιο. Έτσι φτάνουμε στο τελευταίο μπλοκ του κουβά και ελέγχουμε εάν χωράει ακόμα μια εγγραφή. Σε περίπτωση που χωράει, την προσθέτουμε στο τελευταίο μπλοκ. Αντίθετα δεσμεύουμε νέο μπλοκ , αποθηκεύουμε εκεί την εγγραφή και συνδέουμε το νέο μπλοκ με το προηγούμενο που αναφέραμε. Πλέον το νέο μπλοκ είναι το τελευταίο του κουβά. Σε κάθε περίπτωση, ενημερώνουμε κατάλληλα την μεταβλητή που αποθηκεύει το πλήθος των εγγραφών.

HT_DeleteEntry: Εδώ αρχικά διακρίνουμε δύο περιπτώσεις. Η πρώτη περίπτωση είναι το κλειδί να είναι NULL, ενώ η δεύτερη να μην είναι. Στην πρώτη περίπτωση διαγράφουμε την πρώτη εγγραφή μέσα στο αρχείο. Στη δεύτερη, κατακερματίζουμε το κλειδί και πηγαίνουμε στον επιθυμητό κουβά. Έπειτα αναζητούμε την εγγραφή προς διαγραφή μέσα στα μπλοκς του κουβά. Όταν την βρούμε την διαγράφουμε.

Η διαγραφή γίνεται με τον εξής τρόπο:
Το ID της εγγραφής γίνεται 0 και τα υπόλοιπα πεδία-συμβολοσειρές γίνονται “NullString” .
Το παραπάνω επιτυγχάνεται μέσω της βοηθητικής συνάρτησης nullify.

HT_GetAllEntries: Εδώ επίσης διακρίνουμε δύο περιπτώσεις. Εάν το κλειδί είναι NULL ή όχι. Στην πρώτη περίπτωση εκτυπώνουμε σειριακά όλες τις εγγραφές. Στη δεύτερη περίπτωση κατακερματίζουμε το κλειδί και πηγαίνουμε στον αντίστοιχο κουβά. Τέλος, εκτυπώνουμε όλες τις εγγραφές που έχουν ID ίσο με το κλειδί.

HashStatistics: Υλοποιήσαμε δύο βοηθητικές συναρτήσεις ώστε να μπορούμε να βρούμε τα ζητούμενα με μεγαλύτερη ευκολία. Η numRecords, η οποία επιστρέφει το πλήθος των εγγραφών σε ένα κουβά, και η blocksNumber η οποία βρίσκει το πλήθος των μπλοκς σε έναν κουβά. Οι παραπάνω συναρτήσεις χρησιμοποιούν την μέθοδο της αναδρομής. Χρησιμοποιώντας αυτές, η HashStatistics βρίσκει και τυπώνει τα ζητούμενα αποτελέσματα.

Μεταγλώττιση και εκτέλεση του προγράμματος

Η μεταγλώττιση και η εκτέλεση γίνεται με πολύ απλό τρόπο. Δημιουργήσαμε ένα Makefile το οποίο περιέχει τις απαραίτητες εντολές για την μεταγλώττιση του προγράμματος. Επίσης, υπάρχει ένα σύνολο από εντολές για την εκτέλεση του προγράμματος, για όλα τα αρχεία εισόδου που έχουν δοθεί τόσο με αρχείο σωρού, όσο και με αρχείο κατακερματισμού. Για την μεταγλώττιση του προγράμματος, πατάμε την εντολή make.

Για την εκτέλεση του προγράμματος υπάρχουν οι ακόλουθες εντολές:

make run_heap_1K
make run_hash_1K
make run_heap_5K
make run_hash_5K
make run_heap_10K
make run_hash_10K
make run_heap_15K
make run_hash_15K

Για την εκκαθάριση των προσωρινών αρχείων πατάμε την εντολή make clean.

Το πρόγραμμα τρέχει με έναν προκαθορισμένο τρόπο ως προς τα ορίσματα που δέχεται μέσω της γραμμής εντολών. Σε περίπτωση που οι περιορισμοί δεν τηρούνται, το πρόγραμμα εκτυπώνει σχετικό μήνυμα που περιγράφει τον σωστό τρόπο εκτέλεσης και τερματίζει. Οι παραπάνω εντολές που δίνονται μέσω του Makefile τηρούν τους περιορισμούς αυτούς, ώστε το πρόγραμμα να εκτελείται επιτυχώς.