Υλοποίηση Συστημάτων Βάσεων Δεδομένων

Εργασία 1, 2020-2021

Υπεύθυνος Καθηγητής: Κος Δ. Γουνόπουλος

Ομάδα:

Θεοφίλης Κωνσταντίνος - 1115201600287

Κρυπωτός Χρήστος - 1115201700063

Επεξήγηση εργασίας:

Στην εργασία μας ζητείται να υλοποιήσουμε συναρτήσεις που αφορούν την διαχείριση εγγραφών και ευρετηρίων, πάνω από το επίπεδο διαχείρισης μπλοκ. Τα αρχεία που διαχειρίζονται οι συναρτήσεις έχουν δημιουργηθεί βάσει οργάνωσης αρχείου σωρού και στατικού κατακερματισμού. Ακολουθεί εξήγηση των συναρτήσεων.

Αρχικές σημειώσεις:

- Η δομή HT_info είναι ακριβώς όπως προτείνεται στην εκφώνηση, με την απλή προσθήκη του int* hashtable.
- Έχουμε δημιουργήσει μια δομή struct BucketInfo, που περιέχει πληροφορίες για ένα bucket. Συγκεκριμένα, το bitmap του, ένα index για το αν υπάρχει επόμενο bucket (1 αν υπάρχει, -1 αν όχι) και τον αριθμό των εγγραφών που περιέχει.
- Το πρόγραμμα μεταγλωττίζεται με την εντολή make, και εκτελείται με την εντολή ./main.

Υλοποίηση με Πίνακα Κατακερματισμού (Hash Table):

Σε αυτή την υλοποίηση έχουμε τις εξής συναρτήσεις:

HT_CreateIndex: Η συνάρτηση αυτή δημιουργεί ένα καινούργιο αρχείο κατακερματισμού με όνομα filename. Στη συνέχεια δεσμεύει ένα μπλοκ για αυτό το αρχείο και δημιουργεί μια δομή HTinfo που περιέχει τις πληροφορίες του block, απαραίτητες για τις υπόλοιπες συναρτήσεις. Τέλος, τις γράφει στο πρώτο block του αρχείου.

HT_OpenIndex: Η συνάρτηση αυτή ανοίγει ένα αρχείο κατακερματισμού (με την BF_OpenFile), διαβάζει το πρώτο μπλοκ του αρχείου (με την BF_ReadBlock), όπου

περιέχονται οι πληροφορίες που αφορούν το αρχείο. Μετά, τις αποθηκεύει σε μια δομή πληροφοριών την οποία και επιστρέφει.

HT_CloseIndex: Η συνάρτηση διαβάζει το πρώτο block του αρχείου που προσδιορίζεται στη δομή header_info και ελέγχει αν έγινε κάποια αλλαγή σε αυτό. Μετά κάνει overwrite και κλείνει το αρχείο αφού πραγματοποιήσει και τις κατάλληλες αποδεσμεύσεις μνήμης.

HT_InsertEntry: Η συνάρτηση αρχικά ελέγχει το attrType του header_info ώστε να δώσει στην κατάλληλη συνάρτηση κατακερματισμού τον τύπο με βάση τον οποίο θέλουμε να κάνουμε εισαγωγή (δηλ. αν είναι char, βλέπει αν θέλουμε το name, surname κλπ. ή αν είναι int, το id της εγγραφής). Στη συνέχεια, προσπαθεί να εισάγει την εγγραφή. Εδώ μπορεί να υπάρξουν δυο περιπτώσεις:

(α) Να μην έχει γίνει άλλη εγγραφή. Εδώ, η συνάρτηση δημιουργεί ένα νέο block μέσα στο οποίο θα αποθηκεύσει την εγγραφή. Πριν γίνει αυτό, αποθηκεύεται στην πρώτη θέση του νέου block η πληροφορία του. Η εγγραφή αποθηκεύεται στη πρώτη διαθέσιμη θέση.

(β) Να έχει γίνει άλλη εισαγωγή (collision). Εδώ διασχίζει τα buckets μέχρι να βρει την πρώτη διαθέσιμη θέση. Αν την βρει, αποθηκεύει εκεί την εγγραφή. Αν δεν βρει, τότε η συνάρτηση θα πρέπει να δημιουργήσει ένα νέο bucket και να ακολουθήσει την διαδικασία του (α).

HT_DeleteEntry: Η συνάρτηση αρχικά ελέγχει το attrType του header_info, και χρησιμοποιεί την σωστή συνάρτηση κατακερματισμού με τον σωστό τύπο value (int ή char*) ώστε να αποκτήσει το κατάλληλο index. Στη συνέχεια, αναζητεί στα buckets του index την εγγραφή προς διαγραφή είτε μέχρι να την βρει είτε μέχρι να φτάσει στο τελευταίο bucket (δηλαδή το nextBlockIndex = -1). Αν την βρει και είναι valid εγγραφή (bitmap=1), τότε την διαγράφει (αλλάζει το bitmap της θέσης της σε 0 και μειώνει κατά ένα το numRecords του struct BucketInfo και, αποθηκεύει αυτές της αλλαγές στο BucketInfo). Αν εκτελεστεί σωστά, επιστρέφει 0.

HT_GetAllEntries: Η συνάρτηση, όπως και η HT_DeleteEntry, ελέγχει για αρχή το attrType του header_info, ώστε να χρησιμοποιήσει την κατάλληλη συνάρτηση κατακερματισμού με τον κατάλληλο τύπο για την μεταβλητή value και να αποκτήσει το index που προκύπτει. Αφού το αποκτήσει, μπαίνει στα buckets που αντιστοιχούν

στο index και εκτυπώνει όλες τις εγγραφές που έχουν τιμή στο πεδίο-κλειδί ίση με value και είναι valid (έχουν bitmap = 1). Αν υπάρχουν και άλλα buckets, θα συνεχίσει την αναζήτηση σε αυτά μέχρι να φτάσει στο τελευταίο (δηλαδή το nextBlockIndex του να είναι -1). Τέλος, επιστρέφει και το πλήθος block που διαβάστηκαν (block_count).

Υλοποίηση με Αρχείο Σωρού (Heap File):

Στην υλοποίηση με αρχείο σωρού έχουμε τις παρακάτω συναρτήσεις:

HP_CreateFile: Η συνάρτηση αυτή λειτουργεί αρκετά όμοια με την αντίστοιχη της για αρχεία κατακερματισμού. Αρχικά δημιουργεί ένα νέο αρχείο σωρού με όνομα filename και στην συνέχεια αποθηκεύει στο 1^{ο} block του τις απαραίτητες πληροφορίες του (HPInfo).

HP_OpenFile: Η συνάρτηση αυτή ανοίγει το δεδομένο αρχείο filename και διαβάζει από μέσα του τις πληροφορίες του. Μετά τις αποθηκεύει σε μια δομή πληροφοριών την οποία και επιστρέφει.

HP_CloseFile: Η συνάρτηση αυτή διαβάζει το πρώτο block του αρχείου σωρού που προσδιορίζεται στη δομή header_info που δέχεται ως είσοδο και ελέγχει αν έγινε κάποια αλλαγή σε αυτό. Μετά κάνει overwrite και κλείνει το αρχείο αφού πραγματοποιήσει και τις κατάλληλες αποδεσμεύσεις μνήμης.

HP_InsertEntry: Η συνάρτηση αυτή επιδιώκει να εισάγει την εγγραφή που δέχεται ως είσοδο εντός του αρχείου σωρού. Αρχικά ελέγχει αν έχει γίνει πραγματοποιηθεί κάποια εγγραφή στο παρελθόν. Αν δεν έχει ξαναγίνει, δημιουργεί ένα καινούργιο block και αποθηκεύει την πληροφορία του (BucketInfo). Μετά εισάγει την εγγραφή στην πρώτη διαθέσιμη θέση του. Αν έχει ξαναγίνει εγγραφή, τότε αναζητεί αν υπάρχει διαθέσιμη θέση στο block. Αν ναι, τότε αποθηκεύει την εγγραφή εκεί, αλλιώς αν το block έχει τον μέγιστο αριθμό εγγραφών που χωράει, τότε δημιουργεί καινούργιο block και επαναλαμβάνει την διαδικασία της πρώτης περίπτωσης (δεν έχει ξαναγίνει εγγραφή).

HP_DeleteEntry: Η συνάρτηση ελέγχει αρχικά το attrType του header_info ώστε στην συνέχεια να χρησιμοποιήσει το value με τον κατάλληλο του τύπο. Στη συνέχεια, για την τιμή value, η συνάρτηση κοιτάει όλες τις εγγραφές του κάθε bucket του. Αν το

πεδίο κλειδί είναι ίδιο με το value, τότε η εγγραφή διαγράφεται (αλλαγή του bitmap της θέσης σε 0 και μειώνοντας τον συνολικό αριθμό εγγραφών του bucket).

HP_GetAllEntries: Η συνάρτηση ελέγχει για αρχή το attrType του header_info ώστε στην συνέχεια να χρησιμοποιήσει το value με τον κατάλληλο του τύπο. Στη συνέχεια, για την τιμή value, μπαίνει σε κάθε bucket του και εκτυπώνει όλες τις εγγραφές που έχουν τιμή στο πεδίο-κλειδί ίση με value και είναι valid. Τέλος, επιστρέφει και το πλήθος block που διαβάστηκαν (block_count).