Υλοποίηση Συστημάτων Βάσεων Δεδομένων

Ομάδα
A.M : 1115200800090, Χρήστος Δημητριάδης
A.M : 1115201600233, Χρήστος Παπαστεργίου

- **Γενικά** : Στην εργασία έχουν οριστεί για λόγους ευκολίας και καθαρότητας κώδικα μερικές static βοηθητικές συναρτήσεις. Αυτές υπολογίζουν τα offset που χρειαζόμαστε μέσα σε ένα block για να μπορέσουμε να αντλήσουμε ή να πάρουμε κάποια δεδομένα. Στη συνέχεια θα αναλυθούν. Στον κώδικα επίσης υπάρχουν πολλά σχόλια για την επιπλέον κατανόηση του.

Compile & Run

- HeapFile : hp
- HashTable File : ht
- Secondary Hash : sht
- Hash Statistics : stat

Compile : make **one_of_above**
Run : ./build/**one_of_above**_main

Main

- Στις main συναρτήσεις για να γίνει σωστό debugging στην εκτύπωση των ζητούμενων τιμών έχουμε βάλει 5 τιμές που υπάρχουν (μπορεί και όχι ανάλογα το time(NULL)) και μια ακόμα που δεν υπάρχει σίγουρα οπότε βγάζει και το αντίστοιχο μήνυμα. Στο τέλος εκτός από κλείσιμο κάνουμε και delete το αρχείο μας για να μπορεί να τρέξει απροβλημάτιστα ξανά χωρίς να πετάξει το μήνυμα ότι υπάρχει ήδη. Για την main της statistics καλούμε τη συνάρτηση μας με το διαφορετικό όνομα του αρχείου που θέλουμε.

Heap File

- **HP_CreateFile** : Δημιουργούμε το αρχείο μας και περνάμε ένα string μέσα στο πρώτο block σαν το “αναγνωριστικό” του αρχείου μας (όταν ανοίγει το αρχείο να ξέρουμε ότι μιλάμε για heap file).

- **HP_OpenFile** : Όταν έρχεται η ώρα να ανοίξουμε το αρχείο θα πρέπει να αρχικοποιήσουμε και τις τιμές των μεταδεδομένων μας δηλαδή του HP_Info & HP_BlockInfo τα οποία βρίσκονται στο block 0. Για την αντιγραφή στο σωστό σημείο στη μνήμη χρησιμοποιούμε τις συναρτήσεις των offset που υπολογίζουν ουσιαστικά το χώρο που καταλαμβάνει η κάθε δομή μέσα στο block μας ή το string μας.

- **HP_CloseFile** : Εδώ απλά χρησιμοποιούμε την BF_CloseFile συνάρτηση για να κλείσουμε το αρχείο μας.

- **HP_InsertEntry** : Αρχικά ελέγχουμε άν πρέπει να δημιουργηθεί καινούργιο block η όχι (δύο περιπτώσεις : 1. αν είμαστε στην αρχή, 2. άν ένα block έχει γεμίσει τότε δεσμεύουμε για επόμενο). Αυτό το επιτυγχάνουμε με την χρήση μίας μεταβλητής fileBlockSlot που ουσιαστικά σε αυτές τις δύο περιπτώσεις θα είναι 0. Μετά υλοποιούμε τον δείκτη του κάθε block να δείχνει στο επόμενο block. Βέβαια το πρώτο block επειδή περιέχει μόνο μεταδεδομένα θα έχει άλλο offset. Για να μπορέσουμε να δείξουμε τον δείκτη του τωρινού block στον επόμενο ουσιαστικά χρησιμοποιούμε ακόμα ένα προσωρινό block (previousBlock) για να αναφέρεται στο “προηγούμενο”. Ελέγχουμε άν πρέπει να ενημερώσουμε τις τιμές του block (δύο περιπτώσεις : 1. αν είμαστε στην αρχή, 2. άν ένα block είναι καινούργιο πάλι με την ίδια μεταβλητή όπως πριν). Αντιγράφουμε την εγγραφή και κάνουμε dirty τα block μας unpin και destroy.

- **HP_GetAllEntries** : Για να μπορέσουμε να βρούμε όλες τις εγγραφές με το συγκεκριμένο value (id) κάνουμε μία διπλή επανάληψη ώστε να περάσουμε από κάθε block και από κάθε θέση του block για να δούμε άν η εγγραφή είναι αυτή που θέλουμε. Το block_info εφόσον βρίσκεται στο τέλος του block αυτό σημαίνει ότι το offset του είναι μετά από όλες τις εγγραφές που μπορούν να

γίνουν σε ένα block. Επίσης όταν τελείωνουμε με τον έλεγχο αν μια εγγραφή είναι αυτή που θέλουμε πρέπει να αλλάζουν και τη τιμή του record για να μπορεί να πάει στο επόμενο. Επομένως έχουμε και εκεί ένα offset. Κάθε φορά που τελείωνει το κάθε block αυξάνουμε και μια μεταβλητή που στο τέλος θα την επιστρέψουμε και μας λέει πόσα block επισκευτήκαμε μέχρι να βρούμε όλες τις εγγραφές που ψάχναμε. Τέλος πρίν ξεκινήσει νέα επανάληψη για το επόμενο block κάνουμε unpin για να μην έχουμε memory leaks.

Hash File

- **HT_CreateFile** : Δημιουργούμε το αρχείο μας και περνάμε και εδώ ένα string μέσα στο πρώτο block σαν το “αναγνωριστικό” του αρχείου μας (όταν ανοίγει το αρχείο να ξέρουμε ότι μιλάμε για Hashtable file). Εφόσον εδώ μας δίνεται ο αριθμός των buckets μας φάνηκε πιο λογικό να κάνουμε εδώ τις αρχικοποίησεις των τιμών του HT_Info & HT_BlockInfo.

- **HT_OpenFile** : Εφόσον στην createfile κάναμε τις αρχικοποιήσεις τών τιμών του HT_Info & HT_BlockInfo εδώ το μόνο που χρειάζεται να κάνουμε είναι να επιστρέψουμε το HT_Info το οποίο και βάζουμε να δείχνει στο σημείο του block που βρίσκεται δηλάδη μετά το αναγνωριστικό string.

- **HT_CloseFile** : Πάλι χρησιμοποιούμε την BF_CloseFile συνάρτηση για να κλείσουμε το αρχείο μας.

- **HT_InsertEntry** : Για να βάλουμε κάποια καινούργια εγγραφή βρίσκουμε την τιμή της HT_Function για να δούμε σε ποιον κάδο θα μπει η εγγραφή. Ύστερα, μέσω του ht_info ελέγχουμε τον πίνακα hashtable και βρίσκουμε σε ποιο block δείχνει ο κάδος. Αν η τιμή είναι -1 αυτό σημαίνει πως είναι η πρώτη φορά που θα βαλουμε μια τέτοια εγγραφή και έτσι αρχικοποιούμε ένα καινούριο block (μέσω της static συνάρτησης HT_MetadataBlockInitialize) περνάμε την εγγραφή και θέτουμε την τιμή του πίνακα hashtable στη αντίστοιχη θέση ίση με τον αριθμό του block. Στην συνέχεια, εγγραφές με ίδιο ht_hash θα μπούνε στο ίδιο block μέχρι να γεμίσει. Τότε θα πρέπει να αρχικοποιήσουμε ένα καινούριο block, να περάσουμε την εγγραφή, να θέσουμε την τιμή στο hashtable ίση με τον αριθμό του

καινούριου block και να βάλουμε στο block_info του καινούριου block στο πεδίο hashbucket τον αριθμό του block που αντικαταστησαμε στον πίνακα ώστε blocks που αφορούν ίδιο ht_hash να συνδένται μεταξύ τους.

- **HT_GetAllEntries** : Εφόσον έχουμε τον πίνακα κατακερματισμού και κράταμε και τις τιμές των “κοινών” block τότε αρκεί να εξετάζουμε τις τιμές που ψάχνουμε μόνο σε αυτά τα block. Άρα εξετάζουμε ένα πολύ πιο μειωμένο σύνολο block και μέσα σε αυτά τα block τη κάθε τιμη του. Η τιμή hashbucket του κάθε block έχει αρχικοποιηθεί σε -1 (δεν βλέπει κανένα άλλο block). Άρα η επανάληψη θα τελείωσει όταν υπάρξειαυτό το block και αφού εκτυπωθούν (αν πρέπει) δεδομένα του.

Secondary Hash File

- **SHT_CreateFile** : Τα ίδια με την HT_Create με την μόνη προσθήκη ότι κάνουμε open και τα δύο αρχεία για να μην υπάρξει θέμα με το fileDesk των αρχείων.

- **SHT_OpenFile** : Τα ίδια με την HT_Create.

- **SHT_CloseFile** : Πάλι χρησιμοποιούμε την BF_CloseFile συνάρτηση για να κλείσουμε το αρχείο μας.

- **SHT_InsertEntry** : Τα ίδια με την HT_Create απλά επειδή πρέπει να αντιγράψουμε δύο τιμές (1. το όνομα της εγγραφής, 2. το id του block που βρίσκεται) πρέπει να κάνουμε δύο memcpy.

- **SHT_GetAllEntries** : Εδώ αντίστοιχα με την HT_GetAllEntries θα πάρουμε από το Hashtable μας την τιμή και θα κάνουμε αναζήτηση. Απλά όταν βρίσκουμε την εγγραφή με το όνομα θα πρέπει να πάμε στο αντίστοιχο block id του HT αρχείου και να αναζητήσουμε από αυτό όσες εγγραφές έχουν το ίδιο όνομα. Για να μην υπάρχουν διπλότυπα στις εκτυπώσεις έχουμε χρησιμοποιήσει έναν πίνακα (χωρητικότητα όσα και τα block του ht αρχείου) που αρχικοποιούμε τις τιμές του σε 0 και όταν γίνει επίσκεψη μια φορά σε ένα block του HT αρχείου δε θα το ξαναμπεί δεύτερη φορά.

HashStatistics

Αρχικά κάνουμε έλεγχο για το τι αρχείο είναι καθώς θα πρέπει να χρησιμοποιήσουμε διαφορετικά πράγματα για ht και sht. Και πάλι έχουμε ορίσει κάποιες static συναρτήσεις για να μας είναι πιο εύκολη η διαδικασία των offset. Αφού βρούμε τι αρχείο είναι και έχουμε αρχικοποίησει τις μεταβλητές μας ξεκινάει ο κώδικας για να βρούμε τα ζητούμενα. Για τα ελάχιστα-μέγιστα και μέσο όρο έχουμε χρησιμοποιήσει 3 “κανονικές” μεταβλητές και ακόμα 2 που θα είναι οι temporary μεταβλητές ώστε σε κάθε επανάληψη να κρατάμε τα δεδομένα μας. Στην συνέχεια για την υπερχείληση απλά πάμε από block σε block μέσω του hashbucket μας ή αλλιώς ο δείκτης μας σε επόμενο block και τα μετράμε.