Υλοποίηση Συστημάτων Βάσεων Δεδομένων

2η Εργασία
Επεκτατός Κατακερματισμός με Δευτερεύοντα Ευρετήρια

Λεωνίδας Σαρλάς (Α.Μ. 1115201700135)
Απόστολος Θεοδώρου (Α.Μ.: 1115201500046)
Νίκος Κουσουνής (Α.Μ.: 1115201400079)

Χειμερινό Εξάμηνο 2021-2022
Τμήμα Πληροφορικής και Τηλεπικοινωνιών
Εθνικό και Καποδιστριακό Πανεπιστήμιο Αθηνών

Contents

1 Περιγραφή της εργασίας

Σκοπός της παρούσας εργασίας είναι η υλοποίηση προγράμματος σε γλώσσα C για την αποθήκευση εγγραφών σε αρχείο με τη χρήση πρωτεύοντος και δευτερεύοντος ευρετηρίου με πίνακα επεκτατού κατακερματισμού.

Οι εγγραφές έχουν τη μορφή {ID, Name, Surname, City} και προκύπτουν τυχαία από πίνακες ονομάτων και πόλεων που βρίσκονται στη συνάρτηση main.Ο χειρισμός των αρχείων γίνεται σε επίπεδο Block με τη χρήση κατάλληλων συναρτήσεων που δίνονται υπό την μορφή έτοιμης βιβλιοθήκης.

2 Σχεδιασμός - Παραδοχές

Με μια ματιά:

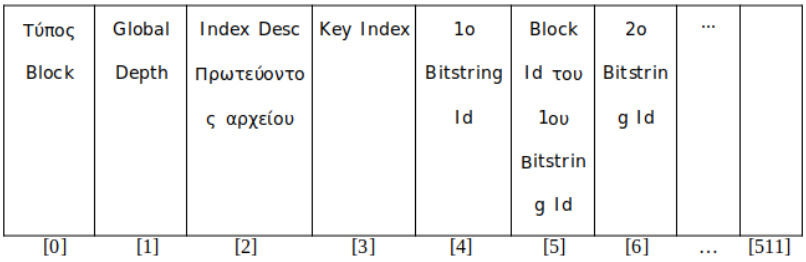
- Οι δομές και ο σχεδιασμός του πρωτεύοντος ευρετηρίου είναι ο ίδιος που είχαμε στην Εργασία 1

Για το δευτερεύον ευρετήριο ισχύει ότι:

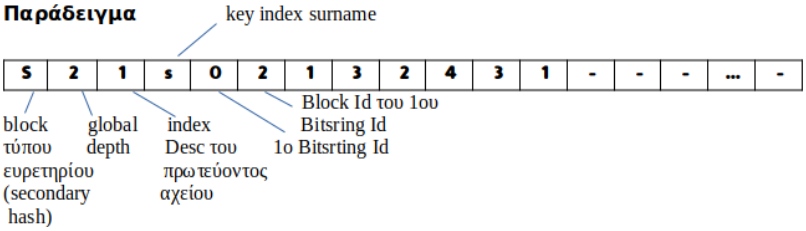
- Κάθε bucket χωράει ένα block
- Ο πίνακας κατακερματισμού αποθηκεύεται σε ένα μόνο block
- Ως συνέπεια του προηγούμενου: κάθε αρχείο δύναται να περιέχει ως 128 blocks (127 για εγγραφές + 1 για το ευρετήριο)
- Για τη δοθείσα main η υλοποίησή μας αποθηκεύει συνολικά 193 + 1 εγγραφές, φτάνοντας το μέγιστο δυνατό αριθμό (21) εγγραφών με κοινό κλειδί (index key, surname)

Κάθε αρχείο αποτελείται από blocks. Τα blocks μπορούν είτε να φυλάσσουν πληροφορία για τον πίνακα κατακερματισμού είτε να φυλάσσουν πραγματικές εγγραφές. Το πρώτο block κάθε αρχείου περιέχει τον πίνακα κατακερματισμού και όλα τα επόμενα είναι blocks εγγραφών. Παρακάτω παρουσιάζεται η δομή ενός block ευρετηρίου και η αντίστοιχη ενός block εγγραφών:

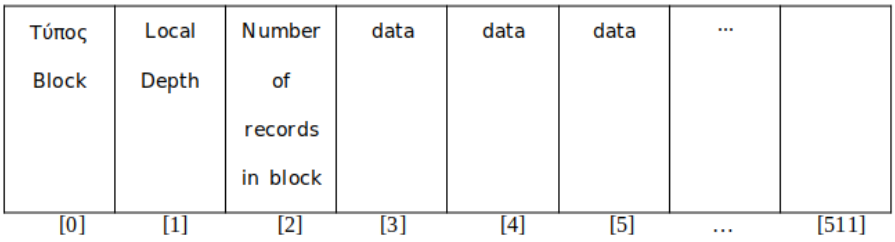
Δομή ενός block δευτερεύοντος ευρετηρίου



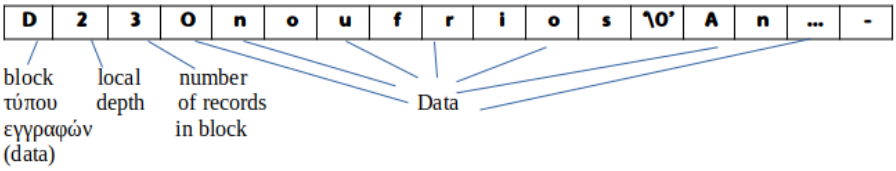
Παράδειγμα

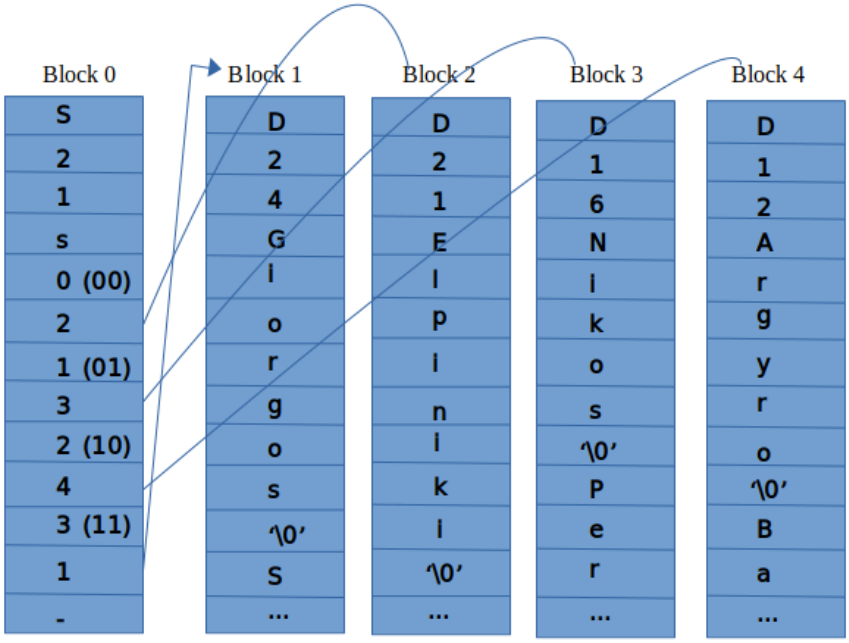


Δομή ενός block εγγραφών



Παράδειγμα





* Κανονικά αρχικά στα blocks δεδομένων πριν το όνομα, προηγείται το record ID

3 Συναρτήσεις - Βασικά Σημεία

Υλοποιήσαμε όλες τις συναρτήσεις που ζητούνται στην εκφώνηση. Παρακάτω περιγράφουμε συνοπτικά τη λειτουργικότητα της κάθε μιας:

- SHT_Init(): δεν κάνει τίποτα γιατί δεν χρειαστήκαμε κάποια παραπάνω δομή
- SHT_CreateSecondaryIndex: Δημιουργεί το δευτερεύον αρχείο επεκτατού κατακερματισμού, δεσμεύει και αρχικοποιεί το πρώτο block του αρχείου (αυτό που φυλάσσεται το ευρετήριο) σύμφωνα με το πρότυπο που φαίνεται πιο πάνω στη σελίδα 2.
- SHT_OpenSecondaryIndex: ανοίγει ένα ήδη υπάρχον αρχείο και ενημερώνει τον πίνακα με τα ανοικτά αρχεία

- SHT_CloseSecondaryIndex: κλείνει ένα ανοικτό αρχείο επεκτατού κατακερματισμού και το αφαιρεί από τον πίνακα των ανοικτών αρχείων

- SHT_SecondaryInsertEntry: Εισάγει μια εγγραφή σε ένα αρχείο ακολουθώντας τα παρακάτω βήματα:

    1. Βρίσκει σε ποιο bucket του πίνακα κατακερματισμού ανήκει (hash value) η εγγραφή (line 194)
    2. Αν είναι η πρώτη εγγραφή που θα μπει στο συγκεκριμένο bucket (το bucket έδειχνε στο 0) δημιουργεί το block και καταχωρείται σε αυτό (line 201-220)
    3. Αν έχει ήδη αρχικοποιηθεί το block ελέγχει το πλήθος των εγγραφών που βρίσκονται σε αυτό και
    4. Αν χωράει, καταχωρείται (line 261)
    5. Αν είναι γεμάτο το block ελέγχει το τοπικό του βάθος και
    6. Αν είναι μικρότερο του global (line 292) δεσμεύουμε ένα νέο block (line 296) βρίσκουμε τον αριθμό των buddies (line 316) βρίσκουμε τον 1ο από αυτούς (line 338) και οι κάνουμε τους πρώτους μισούς να δείχνουν στο παλιό block (line 345) και τους άλλους μισούς στο νέο. Μετά καταχωρούμε αρχικά την νέα εγγραφή (line 362-377) και μετά τις εγγραφές του παλιού block (line 406) (αναδρομικά) στα δύο blocks ανάλογα με το hash value τους.
    7. Αν είναι ίσο με το global (line 421) αποθηκεύουμε προσωρινά τον παλιό πίνακα κατακεμρατισμού, διπλασιάζουμε τον πίνακα, τον αρχικοποιούμε με τις σωστές τιμές bitstring IDs και buckets και στη συνέχεια εφαρμόζουμε τα ίδια με την προηγούμενη περίπτωση.

- SHT_SecondaryUpdateEntry: Δέχεται ως όρισμα έναν πίνακα με τις εγγραφές που ενδεχομένως άλλαξαν bucket κατά τη διαδικασία μιας εισαγωγής στο πρωτεύον ευρετήριο. Εντοπίζει τις εγγραφές αυτές στο δευτερεύον ευρετήριο και ενημερώνει το tupple Id τους

- SHT_PrintAllEntries: Χασάρει το index key , το βρίσκει στο bucket του και το εκτυπώνει

- SHT_HashStatistics: Ομοίως με την HT_PrintAllEntries για NULL , διατρέχει όλα τα bitstring IDs και βρίσκει το μέγιστο και τον ελάχιστο αριθμό εγγραφών του αρχείου, καθώς επίσης και το συνολικό άθροισμα των τοπικών βαθών. Έπειτα καλεί βοηθητική συνάρτηση print_statistics (in file filetable.c) για να εμφανίσει διάφορα στατιστικά στοιχεία

- Στο αρχείο filetable.c υλοποιείται η δομή του πίνακα των ανοικτών αρχείων και κάποιες βοηθητικές συναρτήσεις.

- SHT_InnerJoin: Για συγκεκριμένο index key βρίσκει κάθε εγγραφή του πρώτου αρχείου και του δεύτερου αρχείου με το συγκεκριμένο πεδίο και τυπώνει το καρτεσιανό τους γινόμενο. Για index key= NULL για κάθε μία εγγραφή του πρώτου αρχείου τυπώνει τη ζεύξη της με κάθε όμοια εγγραφή στο δεύτερο αρχείο.

4 Μεταγλώττιση

Εντολή μεταγλώττισης: make
Πως να τρέξετε το πρόγραμμα

- Για την main με τα επίθετα ./build/runner1

- Για την main με τις πόλεις:./build/runner2

Αν θέλετε να το ξανατρέξετε διαγράψτε προηγουμένως όλα τα data.db αρχεία με την εντολή make clean.

Η συνάρτηση SHT_InsertEntry κάνει πολλές τυπώσεις οι οποιές περιγράφουν αναλυτικά την διαδικασία της εισαγωγής μιας εγγραφής. Προτείνεται να τρέξετε το πρόγραμμα για μικρό αριθμό εγγραφών (ht_main1.c line 9 RECORDS_NUM) για να δείτε αναλυτικά την λειτουργικότητά του. Αλλιώς, για μεγαλό πλήθος εγγραφών μπορείτε να δείτε τους πίνακες των στατιστικών που εμφανίζονται στο τέλος της εκτύπωσης.