Υλοποίηση Συστημάτων Βάσεων Δεδομένων

Χειμερινό Εξάμηνο 2020 – 2021

Άσκηση 2

Νικόλας Ηλιόπουλος Α.Μ. 1115201800332

Μιχάλης Βολάκης Α.Μ. 1115201800022

Γενικές Πληροφορίες

```
Κάνουμε BF_init() μόνο μία φορά στην main.
```

Μεταγλώττιση & Εκτέλεση

Εμπεριέχεται Makefile το οποίο μεταγλωττίζει συνολικά όλα τα αρχεία C με την εντολή make. Η εκτέλεση του προγράμματος γίνεται με την εντολή ./prog [-arg1]

- arg1 : Επιλογή αρχείου εγγραφών (1,5,10,15 για records*K.txt)

Δομή Heap File Block

- Bytes 0-480 : Χώρος Εγγραφών (Records)
- Bytes 504-508 : BlockId του επόμενου Block
- Bytes 508-512 : Τρέχων πλήθος εγγραφών στο Block

Δομή Hash File Block

Το Hash File υλοποιείται μέσω Block File. Κάθε block αναπαριστά ένα entry του Hash Table.

Δομή Secondary Hash File Block

Το Secondary Hash File υλοποιείται μέσω Block File. Κάθε block αναπαριστά ένα entry του Secondary Hash Table.

Main Function (main.c)

- Διαβάζει τις εγγραφές από το αρχείο που επιλέχθηκε μέσω του argument (1,5,10,15) και εισάγουμε την εγγραφή για κάθε γραμμή του αρχείου εγγραγών πρώτα στο πρωτεύων HASH TABLE με key το id και έπειτα στο SECONDARY HASH TABLE με key το surname
- Μετά εκτυπώνουμε όλες τις εγγραφές που εισήχθησαν.
- Και εκτυπώνουμε όλα τα στατιστικά και για τα δύο hash table.

Utils (util.c)

Περιέχει βοηθητικές συναρτήσεις για την ανάκτηση/αρχικοποίηση/αλλαγή/εκτύπωση πληροφοριών σε μία εγγραφή ή σε ένα block.

Hashing (hashing.c)

Περιέχει βοηθητικές συναρτήσεις για να δημιουργήσουμε το hash του id μέσω της hash function **SHA1** καθώς και να υπολογίσουμε το αντίστοιχο index στο Hash Table.

Συναρτήσεις HP (Heap File)

- **HP_CreateFile**

Η Create File ελέγχει αρχικά αν υπάρχει το filename που τις δόθηκε μέσω της access() και αν υπάρχει ήδη δεν συνεχίζει στην δημιουργία HP File. Αφότου δημιουργηθεί και ανοιχθεί ένα Block File και γίνει allocate το πρώτο Block αντιγράφουμε τις βασικές πληροφορίες του Heap File σε μία δομή struct firstBlockInfo. Αφότου αντιγράψουμε το struct μέσω της memcpy() στο πρώτο block καλούμε την BF_WriteBlock και κλείνουμε το αρχείο με την BF_CloseFile.

- **HP_OpenFile**

Η OpenFile ανοίγει το Block File, αντιγράφει και επιστρέφει ένα struct HP_info με τις πληροφορίες του πρώτου block. Το Heap File παραμένει ανοιχτό μέχρι την HP_CloseFile.

- **HP_CloseFile**

Η CloseFile κλείνει το heap file και αποδεσμεύει το struct header_info.

- **HP_InsertEntry**

Η InsertEntry αφού ελέγχει την μοναδικότητα του id του Record που δόθηκε διακρίνει 3 περιπτώσεις:

1. *Δεν υπάρχει καμία εγγραφή στο Heap (εκτός του 1ου block πληροφορίας)* : Δεσμέυουμε το δεύτερο block, αντιγράφουμε το record με την memcpy() και αυξάνουμε τον αριθμό των εγγραφών του συγκεκριμένου block.
2. *Το τελευταίο block έχει λιγότερο από 5 εγγραφές* : Μεταφέρουμε τον δείκτη στην τελευταία εγγραφή και αντιγράφουμε το record έχοντας αυξήσει τον αριθμό των εγγραφών.
3. *Το τελευταίο block είναι γεμάτο(περιέχει 5 εγγραφές)* : Με παρόμοιο τρόπο με την (1) δεσμεύουμε νέο block αντιγράφουμε το record και **μεταβάλλουμε το NextNumber του προηγούμενου block** ώστε να δείχνει στο blockId του καινούργιου block.

- **HP_InsertEntrySec**

Η HP_InsertEntrySec κάνει ό,τι κάνει και η HP_InsertEntry απλά εισάγει secondary record αντί για record.

- **HP_DeleteEntry**

Διατρέχει τα entry κάθε block μέχρι να βρει το αντίστοιχο record id και αντικαθιστά τα δεδομένα του record με 0.

- **HP_GetAllEntries**

1. **Περίπτωση value = NULL** : Εκτυπώνουμε όλες τις εγγραφές όλων των block
2. **Περίπτωση value = id** : Εκτυπώνουμε μόνο το record με το αντίστοιχο id(1 εφόσον ελέγξαμε στην insert για μοναδικότητα id).

- **HP_GetAllEntriesSurname**

Η HP_GetAllEntriesSurname είναι ίδια με την HP_GetAllEntries απλά ψάχνει για surname αντί για id.

Συναρτήσεις HT (Hash Table File)

- **HT_CreateIndex**

Η Create Index ελέγχει αν υπάρχει το filename που τις δόθηκε μέσω της access() και αν υπάρχει ήδη δεν συνεχίζει στην δημιουργία HT File. Αφότου δημιουργηθεί και ανοιχθεί ένα Block File και γίνει allocate το πρώτο block αντιγράφουμε τις βασικές πληροφορίες του Hash File σε μία δομή struct firstBlockInfoHT. Αφότου αντιγράψουμε το struct μέσω της memcpy() στο πρώτο block αρχικοποιούμε το κάθε entry(block) του hash table με την δομή hashTableEntry.Καλούμε την BF_WriteBlock και κλείνουμε το αρχείο με την BF_CloseFile.

- **HT_OpenIndex**

Η OpenFile ανοίγει το Block File, αντιγράφει και επιστρέφει ένα struct HT_info με τις πληροφορίες του πρώτου block. Το Hash File παραμένει ανοιχτό μέχρι την HT_CloseFile.

- **HT_CloseIndex**

Η CloseFile κλείνει το heap file για κάθε entry(block) του hash table, κλείνει το ίδιο hash file(bf_CloseFile) και αποδεσμεύει το struct header_info.

- **HT_InsertEntry**

Η InsertEntry χρησιμοποιεί τις συναρτήσεις στην hashing.c για να δημιουργήσει το hash και να υπολογίσει το index του hash table και έπειτα ακολουθεί μία από τις παρακάτω περιπτώσεις:

1. Υπάρχει ήδη bucket(heapExists == 1) άρα καλεί απλά την HP_InsertEntry στο συγκεκριμένο bucket
2. Δεν υπάρχει bucket(heapExists == 0) άρα δημιουργεί heap file αντιγράφει στο συγκεκριμένο entry του hash table πληροφορίες για το heap file και εισάγει το entry μέσω της HP_InsertEntry.

- **HT_DeleteEntry**

Η DeleteEntry υπολογίζει το hashIndex του record id που δόθηκε, μεταβαίνει στο αντίστοιχο entry(block) του hash table
και **μόνο αν υπάρχει bucket** καλεί την HP_DeleteEntry.

- **HT_GetAllEntries**

1. **Περίπτωση value = NULL** : Για κάθε entry του block file **στο οποίο υπάρχει bucket** εκτυπώνουμε όλες τις εγγραφές όλων των block μέσω της HP_GetAllEntries.
2. **Περίπτωση value = id** : Υπολογίζουμε το hashIndex του record id,μεταβαίνουμε στο αντίστοιχο entry(block) και καλούμε την HP_InsertEntry για το συγκεκριμένο id.

- **HashStatistics**

Διατρέχουμε τα blocks του hash table που έχουν bucket και:

α) υπολογίζουμε τον συνολικό αριθμό blocks με τηντotalNumOfBlocks

β) Χρησιμοποιούμε την HP_print_Min_Av_Max_Records (util.c) για να υπολογίσουμε τον ελάχιστο,μέσο και μέγιστο αριθμό εγγραφών που περιέχει το κάθε bucket του αρχείου.

γ) Υπολογίζουμε τον μέσο αριθμό blocks που περιέχουν τα bucket του hash table (totalNumOfBlocks / buckets).

δ) Χρησιμοποιούμε την HP_get_Overflow_Blocks (util.c)για να υπολογίσουμε το πλήθος των overflow buckets και το πόσα blocks είναι αυτά στο κάθε bucket.

Συναρτήσεις SHT (Secondary Hash Table File)

- **SHT_CreateSecondaryIndex**

Η SHT_CreateSecondaryIndex ελέγχει αν υπάρχει το sfilename που τις δόθηκε μέσω της access() και αν υπάρχει ήδη δεν συνεχίζει στην δημιουργία SHT File. Αφότου δημιουργηθεί και ανοιχθεί ένα Block File και γίνει allocate το πρώτο block αντιγράφουμε τις βασικές πληροφορίες του Secondary Hash File σε μία δομή struct firstBlockInfoSHT. Αφότου αντιγράψουμε το struct μέσω της memcpy() στο πρώτο block αρχικοποιούμε το κάθε entry(block) του hash table με την δομή hashTableEntry.Καλούμε την BF_WriteBlock και κλείνουμε το αρχείο με την BF_CloseFile.

- **SHT_OpenSecondaryIndex**

Η SHT_OpenSecondaryIndex ανοίγει το Block File, αντιγράφει και επιστρέφει ένα struct SHT_info με τις πληροφορίες του πρώτου block. Το Secondary Hash File παραμένει ανοιχτό μέχρι την SHT_CloseSecondaryIndex.

- **SHT_CloseSecondaryIndex**

Η SHT_CloseSecondaryIndex κλείνει το heap file για κάθε entry(block) του secondary hash table, κλείνει το ίδιο secondary hash file(bf_CloseFile) και αποδεσμεύει το struct header_info.

- **SHT_SecondaryInsertEntry**

Δεν ελέγχουμε εδώ για μοναδικότητα όπως στο Hash Table γιατί μπορεί να υπάρχουν δύο εγγραφές με το ίδιο επώνυμο.

Η SHT_SecondaryInsertEntry χρησιμοποιεί τις συναρτήσεις στην hashing.c για να δημιουργήσει το hash και να υπολογίσει το index του secondary hash table και έπειτα ακολουθεί μία από τις παρακάτω περιπτώσεις:

1. Υπάρχει ήδη bucket(heapExists == 1) άρα καλεί απλά την HP_InsertEntrySec στο συγκεκριμένο bucket
2. Δεν υπάρχει bucket(heapExists == 0) άρα δημιουργεί heap file αντιγράφει στο συγκεκριμένο entry του hash table πληροφορίες για το heap file και εισάγει το entry μέσω της HP_InsertEntrySec.

- **SHT_SecondaryGetAllEntries**

1. **Περίπτωση value = NULL** : Για κάθε entry του block file **στο οποίο υπάρχει bucket** Καλούμε την HT_GetAllEntries με NULL.
2. **Περίπτωση value = id** : Υπολογίζουμε το hashIndex του secondary record id,μεταβαίνουμε στο αντίστοιχο entry(block) και διατρέχουμε το Bucket ψάχνοντας για το surname που μας ενδιαφέρει, ανακτάμε το block id που δείχνει στο primary index και καλούμε την HP_GetAllEntriesSurname() με ορίσματα την πληροφορία του block file από το bucket του primary index και το surname.

- **HashStatistics**

Ανάλογα με το filename επιλέγουμε τον τρόπο με τον οποίο ελέγχουμε τις εγραφές (+96 bytes για primary index και +32 bytes για secondary index)

Διατρέχουμε τα blocks του hash table που έχουν bucket και:

α) υπολογίζουμε τον συνολικό αριθμό blocks με την totalNumOfBlocks

β) Χρησιμοποιούμε την HP_print_Min_Av_Max_Records (util.c) για να υπολογίσουμε τον ελάχιστο,μέσο και μέγιστο αριθμό εγγραφών που περιέχει το κάθε bucket του αρχείου.

γ) Υπολογίζουμε τον μέσο αριθμό blocks που περιέχουν τα bucket του hash table (totalNumOfBlocks / buckets).

δ) Χρησιμοποιούμε την HP_get_Overflow_Blocks (util.c)για να υπολογίσουμε το πλήθος των overflow buckets και το πόσα blocks είναι αυτά στο κάθε bucket.