Εργασία 2 - Υλοποίηση Συστημάτων Βάσεων Δεδομένων - Δευτερεύον Ευρετήριο & Στατιστικά Κατακερματισμού

Παναγιώτης Δρίβας, A.M.: 1115201900055

Παναγιώτα Γύφτου , A.M.: 1115201900318

Ιανουάριος 2023

Θέμα εργασίας

Υλοποίηση σημαντικών συναρτήσεων που αφορούν τη δημιουργία δευτερεύοντος ευρετηρίου στατικού κατακερματισμού SHT ($Secondary\ Hash\ Table$) και εύρεση στατιστικών πρωτεύοντος και δευτερεύοντος αρχείου.

Παρατήρηση: Το πλήθος των εγγραφών που θα εισαχθούν στο αρχείο είναι μεγέθους 300 $\#define\ RECORDS\_NUM$ 300.

Επιλέχθηκε το 300 για όλα τα είδη οργάνωσης αρχείου, το οποίο κάλλιστα μπορεί να αλλαχθεί.

Τροποποιημένα & Δημιουργημένα Αρχεία

- Αρχείο δευτερεύοντος ευρετηρίου στατικού κατακερματισμού:
  1. *./examples/sht_main.c*
  2. *./include/sht_table.h*
  3. *./src/sht_table.c*
- Στατιστικά κατακερματισμού (δημιουργημένο αρχείο):
  1. *./examples/statistics_main.c*

Μεταγλώττιση & Εκτέλεση

Οι ακόλουθες εντολές λειτουργούν στο αρχικό *directory*, δηλαδή μέσα στον κατάλογο που περιέχει το αρχείο $Makefile$.

- Για να μεταγλωττίσετε και να εκτελέστε το εκτελέσιμο αρχείο,
  - για την υλοποίηση του αρχείου δευτερεύοντος ευρετηρίου στατικού κατακερματισμού, πληκτρολογήστε:

    make sht; **./build/sht_main**

  - για την υλοποίηση των στατιστικών, πληκτρολογήστε:

    make statistics; **./build/statistics_main**

- Για να διαγράψετε τα αρχεία *data.db* και *index.db*, πληκτρολογήστε:

  rm data.db; **rm index.db**

(A) Συναρτήσεις SecondaryHashTable

- **SHT_CreateSecondaryIndex**

Η συνάρτηση $SHT\_CreateSecondaryIndex()$ λειτουργεί παρόμοια με την $HT\_Create\_File()$.
Έγιναν μικρές αλλαγές όπως:

1. Το *struct* των μετα-δεδομένων του αρχείου έγινε από HT_info σε SHT_info και
2. Το *struct* των μετα-δεδομένων κάθε *block* έγινε από HT_block_info σε SHT_block_info.

Οι δύο δομές που δημιουργήθηκαν για χάρη του δευτερεύοντος ευρετηρίου (SHT_info, SHT_block_info), κρατάνε παρόμοιες πληροφορίες με αυτές των αντίστοιχων δομών του πρωτεύοντος ευρετηρίου (HT_info, HT_block_info).

- **SHT_OpenSecondaryIndex**

Η λειτουργία της συνάρτησης $SHT\_OpenSecondaryIndex()$ είναι παρόμοια με αυτή της $HT\_Open\_File()$, με την διαφορά ότι αφαιρέθηκε ο έλεγχος για $heapFile/hashTable$.Όπως στην συνάρτηση $SHT\_CreateSecondaryIndex()$ έγιναν αλλαγές ως προς τις δομές, έτσι και εδώ έγιναν οι ίδιες αλλαγές, δηλαδή,

$$HT\_info \rightarrow SHT\_info \ \& \ HT\_block\_info \rightarrow SHT\_block\_info$$

Τέλος, προστέθηκαν και διάφορες βοηθητικές εκτυπώσεις για την ενημέρωση του χρήστη, όσον αφορά την αρχική κατάσταση του δευτερεύοντος ευρετηρίου.

- **SHT_CloseSecondaryIndex**

Η λειτουργία της συνάρτησης $SHT\_CloseSecondaryIndex()$ είναι ίδια με αυτή της $HT\_Close\_File()$. Η μόνη διαφορά είναι ότι γίνεται ένα λιγότερο $free$, αφού στην υλοποίηση των μετα-δεδομένων μας, μέσω *struct*, δεν χρειαζόμαστε να κρατάμε κάπου ένα *string* με τον τύπο του αρχείου (π.χ. $heapFile/hashTable$).

- **SHT_SecondaryInsertEntry**

Η λειτουργία της συνάρτησης $SHT\_SecondaryInsertEntry()$ είναι παρόμοια με αυτή της $HT\_Insert\_Entry()$.
Έγιναν οι εξής κατάλληλες αλλάγες:

1. $HT\_info \rightarrow SHT\_info$ & $HT\_block\_info \rightarrow SHT\_block\_info$, καθώς και
2. Η αλλαγή του *struct Record* σε $Record\_Secondary$, το οποίο δημιουργούμε μέσα στην συνάρτηση, και μέλος του οποίου αρχικοποιείται με την αξιοποίηση του *record* που μας δόθηκε ως όρισμα.Η δομή $Record\_Secondary$ είναι ορισμένη στο αρχείο $sht\_table.h$.

- **SHT_SecondaryGetAllEntries**

Η λειτουργία της συνάρτησης $SHT\_SecondaryGetAllEntries()$ είναι παρόμοια με αυτή της $HT\_GetAllEntries()$.
Οι διαφορές είναι οι παρακάτω:

1. Όταν βρούμε την σωστή εγγραφή, δεν επιστρέφουμε, αλλά συνεχίζουμε το *search* μέχρι το τελευταίο *block* υπερχείλισης, ώστε να βρούμε και άλλες εγγραφές όμοιες της.

2. Όταν βρούμε εγγραφή με το επιθυμητό όνομα, παίρνουμε το *id* του *block* του πρωτεύοντος αρχείου κατακερματισμού, από το *struct* $Record\_Secondary$, και ελέγχουμε τα ονόματα όλων των εγγραφών του *block* αυτού. Όταν βρούμε το σωστό όνομα, εκτυπώνουμε ολόκληρη την εγγραφή. Το ίδιο κάνουμε και για τα *block* υπερχείλισης.

3. Έχει γίνει η προσθήκη ενός μετρητή, με όνομα $returnValue$, ο οποίος μετράει το συνολικό πλήθος των μπλοκ (πλήθος μπλοκ δευτερεύοντος + πρωτεύοντος ευρετηρίου), που έχουν προσπελαστεί κατά τη διάρκεια εύρεσης των ζητούμενων εγγραφών. Στο τέλος επιστρέφουμε τον αριθμό αυτό.

4. Η τελευταία αλλαγή αφορά την προσθήκη ενός νέου μονοδιάστατου πίνακα $alreadyCheckedBlock$. Κάθε φορά που βρίσκουμε εγγραφή με το επιθυμητό όνομα, ελέγχουμε εάν το id του $block$ του πρωτεύοντος αρχείου κατακερματισμού (από το $struct\ Record\_Secondary$) υπάρχει ήδη στον πίνακα αυτό, δηλαδή, εάν έχει επισκεφτεί αυτό το μπλοκ ξανά στο παρελθόν και έχει ελεγχθεί, εκτυπώνοντας τις επιθυμητές εγγραφές που βρίσκονται σε αυτό. Εάν έχει επισκεφτεί, τότε πηγαίνουμε να ελέγξουμε την επόμενη εγγραφή, διαφορετικά τοποθετούμε το id του μπλοκ αυτού, στον πίνακα και αυξάνουμε το μέγεθος του πίνακα κατά ένα (μέσω της συνάρτησης $realloc$). Το ίδιο κάνουμε και για τις εγγραφές στα $block$ υπερχείλισης.

- **Βοηθητικές Συναρτήσεις**

1. $charHashFunction$:

   Δέχεται ως όρισμα μια συμβολοσειρά για $hash$ (το όνομα ενός $record$) και τον αριθμό των αρχικών $buckets$ και επιστρέφει το id του $block$ που θα αντιστοιχεί στο $record$ αυτό, (επιστρέφει $hashNumber + 1$ διότι το $block$ 0 με τα μετα-δεδομένα πρέπει να αγνοηθεί, και να μην θεωρηθεί ως πιθανό $bucket$ εισαγωγής)

2. $checkIfExists$:

   Ελέγχει εάν ένα $block\_id$ είναι αποθηκευμένο στον πίνακα $alreadyCheckedBlock$, και ανάλογα επιστρέφει μια $boolean$ μεταβλητή.

3. $isSecondaryIndex$:

   Συγκρίνει εάν η συμβολοσειρά που της δόθηκε ως όρισμα είναι ίδια με το όνομα του αρχείου δευτερεύοντος ευρετηρίου.

4. $print\_Metadata\_SecondaryIndex$:

   $Print$ τα μετα-δεδομένα. Χρησιμοποιείται στην συνάρτηση $SHT\_OpenSecondaryIndex$.

5. $print\_SecondaryIndex$:

   Εκτυπώνει το αρχείο δευτερεύοντος ευρετηρίου. Χρησιμοποιείται στις $main()$, των αρχείων $sht\_main.c$ και $statistics\_main.c$, αλλά έχει σχολιαστεί, το χρησιμοποιούμε αποκλειστικά για $debugging$.

- Βοηθητικά structs με τα μέτα-δεδομένα -

1. $struct\ SHT\_info$:

Το $struct$ αυτό κρατάει τα $metadata$ ολόκληρου του αρχείου και γράφεται στην θέση $block$ 0.Περιέχει διάφορα $variables$ τα οποία μας βοηθούν να χειριστούμε το αρχείο κατακερματισμού.Το $block$ αυτό καρφιτσώνεται στην μνήμη με την $openfile$ και παραμένει καρφιτσωμένο μέχρι το κλείσιμο του αρχείου.

Πιο συγκεκριμένα:

fileDesc: Ο αναγνωριστικός αριθμός ανοίγματος αρχείου $index.db$ από το επίπεδο $block$.

InfoAboutHTable Κρατάει δείκτη στο $block$0 με τα μετα-δεδομένα. Χρησιμοποιείται στο $closeFile$, για να κάνουμε το $Block$ αυτο $unpin$.

HashTable: Πίνακας κατακερματισμού. Πρώτη στήλη[$bucketId$]: κρατάει το id των αρχικών $Buckets$ πριν την υπερχείλιση, Δεύτερη στήλη[$lastBucketWithFreeSpace$]: Για κάθε τέτοιο αρχικό

Bucket δείχνει το τελευταίο *block*-υπερχείλισης με ελεύθερο χώρο.
Αρχικά $bucketId == lastBucketWithFreeSpace$.

recordSize: Η χωρητικότητα μιας εγγραφής.

numberOfRecordsInBlock: Η χωρητικότητα ενός *block* σε εγγραφές.

numOfBuckets: Το πλήθος των αρχικών "κάδων" του αρχείου κατακερματισμού. Δίνεται στην $SHT\_CreateSecondaryIndex$ μέσω της *main*.

2. *struct* SHT_block_info:

   Το *struct* αυτό είναι ίδιο με το αντίστοιχο της HT (HT_block_info).

3. *struct Record_Secondary*:

Το *struct* αυτό είναι η δομή των νέων *record* μας. Έχει δύο πεδία, το *charname*[15], το οποίο είναι το όνομα της εγγραφής και το *int blockId* το οποίο είναι το *id* του *block* στο πρωτεύον αρχείο κατακερματισμού που είναι αποθηκευμένη ολόκληρη η εγγραφή.

(B) Συναρτήσεις Στατιστικών

Για τον έλεγχο, των στατιστικών πληροφοριών των δύων ειδών ευρετηρίου, έχει υλοποιηθεί μια συνάρτηση *main*() στο αρχείο statistics_main.c, το οποίο βρίσκεται στον φάκελο *./examples*. Πιο συγκεκριμένα η *main*() συνάρτηση είναι ακριβώς η ίδια με αυτή που βρίσκεται στο αρχείο *sht_main.c* (όπου ελέγχεται η λειτουργικότητα του δευτερεύοντος αρχείου), με την προσθήκη της κλήσης, της συνάρτησης **HashStatistics**. Η *HashStatistics*() καλείτε δύο φορές, μία για χάρη του πρωτεύοντος ευρετηρίου και μία για το δευτερεύον ευρετήριο, ώστε να βρεθούν τα στατιστικά τους.

Η *HashStatistics*() είναι υλοποιημένη και αυτή στο αρχείο *statistics_main.c*, (ακριβώς πάνω από την *main*()). Δουλειά της είναι, ανάλογα με το είδος του ευρετηρίου να καλεί την κατάλληλη συνάρτηση εύρεσης στατιστικών. Πιο συγκεκριμένα η *HashStatistics*(), ελέγχει με την βοήθεια των συναρτήσεων *isSecondaryIndex*() και *isPrimaryIndex*(), εάν το όνομα του αρχείου, που περάστηκε ως όρισμα, είναι δευτερεύον ή πρωτεύον αρχείο ευρετηρίου αντίστοιχα.Στην περίπτωση που το αρχείο είναι αρχείο δευτερεύοντος ευρετηρίου, καλείται η συνάρτηση *HashStatistics_Secondary*(), διαφορετικά εάν το εξετάσιμο αρχείο είναι αρχείο πρωτεύοντος ευρετηρίου τότε καλείται η *HashStatistics_Primary*().

Οι συναρτήσεις εύρεσης στατιστικών *HashStatistics_Secondary*() και *HashStatistics_Primary*(), είναι υλοποιημένες στα αρχεία *sht_table.c* και *ht_table.c* αντίστοιχα.Η υλοποίηση και των δύο είναι ακριβώς η ίδια, απλώς διαφέρουν οι δομές πληροφοριών που χρησιμοποιούν.Αυτή η διαφορά είναι ήδη γνωστή, καθώς έχει αναφερθεί στις περιγραφές των συναρτήσεων του δευτερεύοντος ευρετηρίου.Πιο συγκεκριμένα,

- η *HashStatistics_Secondary*() χρησιμοποιεί τις δομές: SHT_info & SHT_block_info, και
- η *HashStatistics_Primary*() χρησιμοποιεί τις δομές: HT_info & HT_block_info

Η λειτουργία τους είναι οι εξής:

Αρχικά με την βοήθεια της $BF\_OpenFile$(), επιτυγχάνεται η ανάκτηση του περιγραφέα αρχείου, ώστε να χρησιμοποιηθεί για την λειτουργία σημαντικών βοηθητικών συναρτήσεων εύρεσης πληροφοριών του αρχείου.Μία από αυτές είναι η συνάρτηση $BF\_GetBlockCounter$(), μέσω της οποίας λαμβάνουμε το πλήθος των μπλοκ που βρίσκονται στο αρχείο. Στην συνέχεια ελέγχουμε έναν-έναν τους κάδους ώστε να εξάγουμε τις πληροφορίες όσον αφορά το ελάχιστο, το μέσο και το μέγιστο πλήθος εγγραφών που έχει κάθε *bucket* ενός αρχείου, καθώς και το πλήθος των *buckets* που έχουν μπλοκ υπερχείλισης, και πόσα μπλοκ είναι αυτά για κάθε *bucket*. Πιο αναλυτικά κατά τον έλεγχο του κάδου, εξετάζεται:

1. Εξετάζεται στην αρχή το αρχικό μπλοκ που είχε αντιστοιχηθεί σε αυτόν τον κάδο.Εάν πλεον σε αυτόν τον κάδο

δεν είναι αυτό το αρχικό μπλοκ τότε αυξάνεται ο μετρητής που μητράει πόσα *bucket* έχουν υπερχειλίσει.

2. Εξετάζονται και ενημερώνονται οι μεταβλητες που κρατάνε την πληροφορία του ελάχιστου και του μέγιστου πλήθους εγγραφών ($min,\ max$), ανάλογα αν ικανοποιούνται οι συνθήκες με το πλήθος εγγραφών του αρχικού μπλοκ.Καθώς αυξάνεται και ο μετρητης των συνολικών εγγραφών.

3. Εάν υπάρχουν μπλοκ υπερχείλισης εξετάζονται και αυτά.Κατα τον έλεγχο των μπλοκ υπερχείλισης, αυξάνεται ένας μετρητής ο οποίος μετράει τα μπλοκ υπερχείλισης για το τρέχων εξετάσιμο *bucket*, και όπως και στο αρχικό μπλοκ ενημερώνονται, ανάλογα την ικανοποίηση των συνθηκών με το πλήθος εγγραφών του τρέχοντος μπλοκ υπερχείλισης, οι μεταβλητές ($min,\ max$).Καθώς αυξάνεται και ο μετρητης των συνολικών εγγραφών.Μετά την ολοκλήρωση των ελέγχων των μπλοκ υπερχείλισης, αν αυτά υπάρχουν, τότε εκτυπώνεται η πληροφορία για τον τρέχοντα κάδο, το πλήθος των μπλοκ υπερχείλισης και έπειτα ο μετρητής των μπλοκ υπερχείλισης μηδενίζεται για μελλοντική χρήση.

Όταν ολοκληρωθεί και ο έλεγχος όλων των κάδων τότε εκτυπώνεται το πλήθος των *buckets* που έχουν μπλοκ υπερχείλισης.Επιπλέον το ελάχιστο, το μέσο και το μέγιστο πλήθος εγγραφών που έχει κάθε *bucket* ενός αρχείου. Τέλος υπολογίζεται και εκτυπώνεται ο μέσος αριθμός των μπλοκ που έχει κάθε *bucket*.