Άγγελος Τσιτσόλη – 1115202000200

Χριστοφιλοπούλου Βασιλική - 1115202000216

ΥΛΟΠΟΙΗΣΗ ΣΥΣΤΗΜΑΤΩΝ ΒΑΣΕΩΝ ΔΕΔΟΜΕΝΩΝ – ΕΡΓΑΣΙΕΣ 1 & 2

Εργασία 1:

Heap File

Υλοποιήθηκαν οι παρακάτω συναρτήσεις:

- HP_CreateFile : Δημιουργεί και αρχικοποιεί ένα αρχείο σωρού με όνομα fileName. Στο block 0 αποθηκεύουμε τις εξής πληροφορίες:
  1. fileDesc: αριθμός ανοίγματος αρχείου από το επίπεδο block
  2. id: 0 εάν πρόκειται για αρχείο σωρού και 1 εάν πρόκειται για αρχείο στατικού κατακερματισμού.
  3. NumRecords: ο αριθμός εγγράφων που έχει το αρχείο.
  4. IdLast: ο αριθμός του τελευταίου block

- HP_OpenFile: Φτιάχνει μια δομή τύπου HP_info, την αρχικοποιεί και την επιστρέφει στην καλούσα συνάρτηση.

- HP_CloseFile: Αποδεσμεύει την μνήμη της δομής που δημιουργεί η HP_OpenFile και κλείνει το αρχείο σωρού.

- HP_InsertEntry: Γίνεται εισαγωγή μιας εγγραφής στο τέλος του σωρού. Υπάρχουν τρεις περιπτώσεις:
  1. Ο σωρός να είναι άδειος, οπότε δημιουργούμε το πρώτο Block και εισάγουμε την εγγραφή.

2. Να υπάρχει Block αλλά να είναι γεμάτο, οπότε δημιουργούμε νέο block, συνδέουμε κατάλληλα του δείκτες μεταξύ των δύο Block και έπειτα αποθηκεύουμε την εγγραφή.
3. Γενική περίπτωση όπου τοποθετούμε στο υπάρχον Block διότι δεν έχει γεμίσει.

Και στις τρεις περιπτώσεις τροποποιούμε τα αντίστοιχα hp_block_info και hp_info, προκειμένου να είναι πάντοτε πλήρως ενημερωμένα.

- HP_GetAllEntries: Με βάση την τιμή που δίνεται ως όρισμα στην συνάρτηση, αναζητούμε σειριακά τον σωρό προκειμένου να ελέγξουμε εάν υπάρχει η εγγραφή ή όχι στον σωρό. Εφόσον βρεθεί επιστρέφουμε τον αριθμό των Block που χρειάστηκαν να διαβαστούν διαφορετικά επιστρέφει -1.

Hash Table

Υλοποιήθηκαν οι παρακάτω συναρτήσεις:

- HT_CreateFile : Δημιουργεί και αρχικοποιεί ένα πρωτεύον αρχείο στατικού κατακερματισμού με όνομα fileName. Στο block 0 αποθηκεύουμε τις εξής πληροφορίες:
  1. fileDesc: αριθμός ανοίγματος αρχείου από το επίπεδο block
  2. id: 0 εάν πρόκειται για αρχείο σωρού και 1 εάν πρόκειται για αρχείο στατικού κατακερματισμού.
  3. numBuckets: ο αριθμός των «κάδων» του αρχείου κατακερματισμού.
  4. keyAttribute: Το κλειδί με βάση το οποίο θα γίνει ο κατακερματισμός.
  5. NumRecords: Ο μέγιστος αριθμός εγγραφών που μπορούν να αποθηκευτούν σε ένα block.

  Επίσης, στο block 1 δημιουργούμε ένα ευρετήριο και δεσμεύουμε μνήμη για άλλα n blocks, όπου n ο αριθμός των «κάδων» που δίνεται ως όρισμα. Σε περίπτωση που κάποιο block χρειαστεί block υπερχείλισης, η δέσμευση και η αρχικοποίηση του θα γινούν από την συνάρτηση insert_to_bucket.

- HT_OpenFile: Φτιάχνει μια δομή τύπου HT_info, την αρχικοποιεί και την επιστρέφει στην καλούσα συνάρτηση.

- HT_CloseFile: Αποδεσμεύει την μνήμη της δομής που δημιουργεί η HT_OpenFile και κλείνει το κατακερματισμού.

- HT_InsertEntry: Γίνεται εισαγωγή μιας εγγραφής στο πρωτεύον αρχείο κατακερματισμού. Αρχικά, ελέγχουμε ποιο είναι το κλειδί με βάση το οποίο έχει γίνει ο κατακερματισμός και υπολογίζουμε το hash value για να δούμε σε ποιον «κάδο» πρέπει να το τοποθετήσουμε. Τώρα που ξέρουμε που πρέπει να τοποθετηθεί, καλούμε την συνάρτηση insert_to_bucket, η όποια βρίσκει το πρώτο Block του ζητούμενου bucket και γίνεται η εισαγωγή με βάση τις ακόλουθες περιπτώσεις:

    1. Δεν υπάρχει ακόμα «κάδος» υπερχείλισης και η εισαγωγή γίνεται στο πρώτο block.
    2. Το block είναι γεμάτο και αναζητούμε το αμέσως επόμενο πού δεν είναι γεμάτο με σκοπό να γίνει η εισαγωγή.

Και στις δύο παραπάνω περιπτώσεις ελέγχουμε εάν μετά την εισαγωγή έχει γεμίσει το block προκειμένου να δημιουργήσουμε νέο block υπερχείλισης.

Επιπλέον, τροποποιούμε τα αντίστοιχα ht_block_info και ht_info,για να είναι πάντοτε πλήρως ενημερωμένα.

- HP_GetAllEntries: Με βάση την τιμή που δίνεται ως όρισμα στην συνάρτηση και το ποιο είναι το κλειδί πεδίο, βρίσκουμε το hash value για να δούμε σε ποιον «κάδο» βρίσκεται η εγγραφή που αναζητούμε. Συνεπώς ελέγχουμε τον αντίστοιχο «κάδο» και τα αντίστοιχα overflow buckets που έχει. Εφόσον βρεθεί εμφανίζουμε τον αριθμό των Block που χρειάστηκαν να διαβαστούν διαφορετικά εμφανίζουμε μήνυμα ενημέρωσης.

Εργασία 2:

Secondary Hash Table

Υλοποιήθηκαν οι παρακάτω συναρτήσεις:

- SHT_CreateSecondaryIndex: Ακολουθεί την ίδια λογική με την HT_CreateFile, δηλαδή δημιουργεί και αρχικοποιεί ένα δευτερεύον αρχείο στατικού κατακερματισμού με όνομα sfileName. Στο block 0 αποθηκεύουμε τις εξής πληροφορίες:
  1. fileDesc: αριθμός ανοίγματος αρχείου από το επίπεδο block
  2. id: 0 εάν πρόκειται για αρχείο σωρού και 1 εάν πρόκειται για αρχείο στατικού κατακερματισμού.
  3. numBuckets: ο αριθμός των «κάδων» του αρχείου κατακερματισμού.
  4. keyAttribute: Το κλειδί με βάση το οποίο θα γίνει ο κατακερματισμός.
  5. NumPairs: Ο μέγιστος αριθμός εγγραφών που μπορούν να αποθηκευτούν σε ένα block.

  Επίσης, στο block 1 δημιουργούμε ένα ευρετήριο και δεσμεύουμε μνήμη για άλλα n blocks, όπου n ο αριθμός των «κάδων» που δίνεται ως όρισμα. Σε περίπτωση που κάποιο block χρειαστεί block υπερχείλισης, η δέσμευση και η αρχικοποίηση του θα γινούν από την συνάρτηση insert_to_bucket2.

  Επιπλέον, δημιουργήθηκε μια δομή STH_pair, η οποία θα είναι τα στοιχεία που θα περιέχει το δευτερεύον ευρετήριο. Πρόκειται για ζευγάρια δεδομένων εκ των οποίων το ένα είναι το κλειδί με βάση το οποίο γίνεται ο κατακερματισμός και το in_block που είναι ο αριθμός του block που βρίσκεται το στοιχείο στο πρωτεύον ευρετήριο.

- SHT_OpenSecondaryIndex: Φτιάχνει μια δομή τύπου SHT_info, την αρχικοποιεί και την επιστρέφει στην καλούσα συνάρτηση.

- SHT_CloseSecondaryIndex: Αποδεσμεύει την μνήμη της δομής που δημιουργεί η SHT_OpenFile και κλείνει το δευτερεύον κατακερματισμού.

- SHT_SecondaryInsertEntry: Γίνεται εισαγωγή μιας εγγραφής στο δευτερεύον αρχείο κατακερματισμού. Αρχικά, ελέγχουμε ποιο είναι το κλειδί με βάση το οποίο έχει γίνει ο κατακερματισμός και υπολογίζουμε το hash value για να δούμε σε ποιον «κάδο» πρέπει να το τοποθετήσουμε. Δημιουργείται το SHT_pair που είναι το δεδομένο που θα περιέχει το δευτερεύον ευρετήριο και δίνεται ως όρισμα στην συνάρτηση insert_to_bucket2, η όποια βρίσκει το πρώτο Block του ζητούμενου bucket και γίνεται η εισαγωγή με βάση τις ακόλουθες περιπτώσεις:

  1. Δεν υπάρχει ακόμα «κάδος» υπερχείλισης και η εισαγωγή γίνεται στο πρώτο block.
  2. Το block είναι γεμάτο και αναζητούμε το αμέσως επόμενο πού δεν είναι γεμάτο με σκοπό να γίνει η εισαγωγή.

Και στις δύο παραπάνω περιπτώσεις ελέγχουμε εάν μετά την εισαγωγή έχει γεμίσει το block προκειμένου να δημιουργήσουμε νέο block υπερχείλισης.

Επιπλέον, τροποποιούμε τα αντίστοιχα sht_block_info και sht_info,για να είναι πάντοτε πλήρως ενημερωμένα.

- SHT_SecondaryGetAllEntries: Με βάση την τιμή που δίνεται ως όρισμα στην συνάρτηση και το ποιο είναι το κλειδί πεδίο, βρίσκουμε το hash value για να δούμε σε ποιον «κάδο» βρίσκεται η εγγραφή που αναζητούμε. Συνεπώς ελέγχουμε τον αντίστοιχο «κάδο» και τα αντίστοιχα overflow buckets που έχει. Εφόσον βρεθεί, αντλούμε από το SHT_pair τον αριθμό του Block του πρωτεύοντος ευρετηρίου βρίκεται η ζητούμενη εγγραφή. Φορτώνουμε στην μνήμη αυτό το block και ελέγχουμε σειριακά τις εγγραφές του εως ότου βρεθεί εκείνη που μας ενδιαφέρει. Εκτυπώνουμε τα πεδία της εγγραφής και εμφανίζουμε τον αριθμό των Block που χρειάστηκαν να διαβαστούν διαφορετικά εμφανίζουμε μήνυμα ενημέρωσης.

Δημιουργήθηκαν επίσης οι βοηθητικές συναρτήσεις: MinTable, MaxTable και Sum οι οποίες βρίσκουν το ελάχιστο στοιχείο, το μέγιστο στοιχείο ή το άθροισμα ενός πίνακα.

Σχόλια:

- Το αναγνωριστικό fileDesc αρχικοποιείται τόσο για το πρωτεύον όσο και για το δευτερεύον ευρετήριο στην open αντί να γίνεται στην create για να μπορούν να διαχωριστούν μεταξύ τους.

- Στις create των ευρετηρίων έχει προστεθεί σαν όρισμα Record_Attribute keyAttribute, έτσι ώστε να μπορούν να δημιουργηθούν και ευρετήρια με κλειδί κάποιο άλλο χαρακτηριστικό.

- Δίνεται μαζί με τους αντίστοιχους κώδικές και το αρχείο MakeFile στο όποιο έχουν προστεθεί οι αντίστοιχες εντολές προκειμένου να εκτελείται σωστά και η main που δημιουργήσαμε εμείς.

Για την μεταγλώττιση: make main

Για την εκτέλεση: ./build/our_main