Εργασία 2

Συντελεστές και Αριθμοί Μητρώου

Αντώνιος Κληρονόμος, 1115201500069
Λάζαρος Ζερβός, 1115201500042

Γλώσσα Υλοποίησης
C

Περιγραφή Υλοποίησης

Δομή των blocks:

- Δομή του block που περιέχει τα metadata του αρχείου: Περιέχει τα "This is a B+ tree file", τον τύπο του πρώτου πεδίου των εγγραφών, το μήκος του πρώτου πεδίου των εγγραφών, τον τύπο του δεύτερου πεδίου των εγγραφών, το μήκος του δεύτερου πεδίου των εγγραφών καθώς και το block id της ρίζας του B+ δένδρου, διαχωριζόμενα από "$".

- Δομή του data block: Χρησιμοποιούμε για αναγνώριση των data blocks στην αρχή του block το ":)". Έπειτα σαν πρώτη "εγγραφή" έχουμε κάποια metadata που χρειαζόμαστε. Συγκεκριμένα έχουμε το block id του block-γονέα, το πλήθος των εγγραφών στο block και το block id του επόμενου block δεδομένων. Γενικά διαχωρίζουμε τα πεδία των εγγραφών με "$" και τις εγγραφές μεταξύ τους με "&". Οι πραγματικές εγγραφές και οχι τα metadata αποτελούνται απο τα value1,value2 όπου value1 το κλειδί της εγγραφής και value2 το πεδίο της.
  Παράδειγμα data block: :)&3$12$5&252$antonis&156$giorgos&...

- Δομή του index block: Αντίστοιχα στα index blocks δεν έχουμε αναγρνωριστικό(ετσι τα ξεχωρίζουμε απο τα data blocks τα οποία έχουν). Όπως και στα data blocks τα πεδία των εγγραφών διαχωρίζονται με "$" ενώ οι εγγραφές με "&". Τα metadata είναι το block id του block-γονέα και το πλήθος των εγγραφών στο block(P0, K1, P1, K2, P2, ...). Γενικά κάθε εγγραφή (εκτός από την πρώτη που είναι μόνο ένας pointer(P0)(δηλαδή ένα block id)) αποτελείται απο ένα κλειδι(Ki)(δηλαδή το value1) και έναν pointer(Pi)(δηλαδή ένα block id) που δείχνει στο block δεδομένων-παιδί που έχει εγγραφές με κλειδί μεγαλύτερο από Ki και μικρότερο από K(i+1), **εάν βρισκόμαστε στο προτελευταίο επίπεδο**. **Εάν βρισκόμαστε σε ανώτερο επίπεδο**, ο Pi δείχνει σε block ευρετηρίου με κλειδιά εγγραφών μεγαλύτερα ή ίσα του Ki και μικρότερα ή ίσα του K(i+1).
  Παράδειγμα index block: 2$14&P0&K1$P1&K2$P2&...

Συναρτήσεις:

void sort2arrays(char array1, char** array2, int n, File curr_file)**
Είναι μια συνάρτηση που υλοποιεί τον αλγόριθμο ταξινόμισης selection sort για να ταξινομήσει παράλληλα τους πίνακες array1 και array2, κατά αύξουσα σειρά των τιμών του πίνακα array1. Χρησιμοποιείται για την ταξινόμιση των εγγραφών σε data blocks.

void sort2arraysInt(char array1, int* array2, int n, File curr_file)**
Αυτή η συνάρτηση έχει την ίδια λειτουργία με την προηγούμενη απλώς σαν array2 δέχεται έναν πίνακα από int. Χρησιμοποιείται για την ταξινόμιση των εγγραφών σε index blocks.

int compareKeys(File curr_file, char* value, char* value1)
Συγκρίνει δύο τιμές κλειδιών (value και value1) ανάλογα με τον τύπο των κλειδιών του αρχείου curr_file. Επιστρέφει -1 όταν value < value1, επιστρέφει 1 όταν value > value1 και 0 όταν value = value1.

int findSuitableBlockId(int blockId, void* value1, File* curr_file, int fileDesc)
Η συνάρτηση αυτή είναι αναδρομική και προσπελαύνει το B+ δένδρο ξεκινώντας από το block με id blockId (που όταν καλείται για πρώτη φορά είναι το id της ρίζας του δένδρου). Συγκρίνοντας σταδιακά το κλειδί value1 με τα κλειδιά Ki των index blocks, καταλήγει στο data block στο οποίο η εγγραφή με κλειδή value1 πρέπει κανονικά να μπει και επιστρέφει το block id του, χωρίς να ελέγχει αν χωράει ή να την βάζει. Αυτές οι διαδικασίες πραγματοποιούνται αργότερα από άλλες συναρτήσεις.

int split(int blockId, int fileDesc, File* curr_file, void* value1, void* value2, char* indexValue, int childBlockId)
Γενικά, εφαρμόζει την τεχνική του "σπασίματος" στα B+ δενδρα. Πιο συγκεκριμένα, είναι αναδρομική και δέχεται ως όρισμα ένα blockId το οποίο αρχικά είναι id ενός data block το οποίο πρέπει να "σπάσει" για να εισαχθεί η εγγραφή (value1, value2).

Εάν το blockId είναι id ενός data block:

- Εάν αυτό το data block δεν έχει πατέρα-index block, το δημιουργεί και μεταφέρει το key(value1) της "μεσαίας" εγγραφής αυτού του data block και το id του στον πατέρα-indexblock. Έπειτα χωρίζει το data block σε δύο data blocks με "μισό" αριθμό εγγραφών το καθένα (χρησιμοποιείται το "/" (div) για τη διαίρεση).
- Εάν αυτό το data block έχει πατέρα-index block, η συνάρτηση καλεί τον εαυτό της βάζοντας το key της "μεσαίας" εγγραφής αυτού του data block στο όρισμα indexValue και το block id του στο childBlockId. Επίσης ως blockId βάζει το block id του πατέρα-index block. Όταν επιστρέψει, χωρίζει το data block σε δύο data blocks με "μισό" αριθμό εγγραφών το καθένα (χρησιμοποιείται το "/" (div) για τη διαίρεση).

Εάν το blockId είναι id ενός index block:

- Εάν η εγγραφή (indexValue, childBlockId) χωράει στο block, την εισάγει.
- Εάν η εγγραφή (indexValue, childBlockId) δε χωράει στο block, τότε
  - Εάν αυτό το index block δεν έχει πατέρα-index block, τότε το δημιουργεί και μεταφέρει το key(Ki) της "μεσαίας" εγγραφής αυτού του index block και το block id(Pi) που βρίσκεται δεξιά αυτού του key στον πατέρα-index block. Έπειτα χωρίζει το index block σε δύο index blocks με "μισό" αριθμό εγγραφών το καθένα (χρησιμοποιείται το "/" (div) για τη διαίρεση) χωρίς όμως να συμπεριλαμβάνει το ζευγάρι Ki, Pi που μεταφέρθηκε στον πατέρα-index block.
  - Εάν αυτό το index block έχει πατέρα-index block, η συνάρτηση καλεί τον εαυτό της βάζοντας το key(Ki) της "μεσαίας" εγγραφής αυτού του index block στο όρισμα indexValue και το block id(Pi) που βρίσκεται δεξιά του key στο childBlockId. Επίσης ως blockId βάζει το block id του πατέρα-index block. Όταν επιστρέψει, χωρίζει το index block σε δύο index blocks με "μισό" αριθμό εγγραφών το καθένα (χρησιμοποιείται το "/" (div) για τη διαίρεση) χωρίς όμως να συμπεριλαμβάνει το ζευγάρι Ki, Pi που μεταφέρθηκε στο index block-πατέρα.

Καθόλη τη διάρκεια της παραπάνω διαδικασίας, τα metadata όλων των blocks που εμπλέκονται σε αυτήν ενημερώνονται κατάλληλα και οι εγγραφές όταν βγαίνουν και ξαναμπαίνουν στα blocks ταξινομούνται σε αύξουσα σειρά, ανάλογα με τον τύπο τους, με βάση το πεδίο-κλειδί τους.

int AM_InsertEntry(int fileDesc, void *value1, void *value2)
Αναλαμβάνει την εισαγωγή της εγγραφής (value1, value2) σε συνεργασία με τη **findSuitableBlockId** και τη **split**. Συγκεκριμένα:

- Εάν υπάρχει μόνο ένα block στο αρχείο (metadata-block) δημιουργεί ένα data block στο οποίο εισάγει τη δεδομένη εγγραφή.
- Αλλιώς, καλεί τη **findSuitableBlockId** με πρώτο όρισμα τη ρίζα του B+ δένδρου ώστε να βρει σε ποιο block πρέπει να μπεί κανονικά η εγγραφή.
  - Εάν η εγγραφή χωράει στο data block του οποίου το id επέστρεψε η **findSuitableBlockId**, την εισάγει και ταξινομεί όλες τις εγγραφές του block
  - Αλλιώς, καλεί τη συνάρτηση **split** η οποία "σπάει" το data block και όσα άλλα index blocks χρειάζεται και τελικά εισάγει την εγγραφή

int AM_OpenIndexScan(int fileDesc, int op, void *value)
Σε αυτή την συνάρτηση θέλουμε να πάρουμε τα αποτελέσματα από κάποιο "query" και να τα εμφανίσουμε 1-1 καλώντας την Find_NextEntry αντίστοιχες φορές. Με διάφορα strtoκ(κυρίως) και strchr παίρνουμε τα metadata και τις εγγραφές απο κάθε block προσπελαύνοντας τα,κατάλληλα.Διαδικασία προσπέλασης των index blocks: αν ο operator είναι ο EQUAL τότε πάμε στο κατάλληλο block ενώ αν είναι κάποιος άλλος τότε πάμε στο πιο αριστερό φύλλο,(πηγαίνοντας σε κάθε κόμβο(index block) στο αριστερό παιδί του) και από εκεί προσπελαύνουμε τα data blocks. Στα data blocks, εκτός από κάποιες περιπτώσεις, ελέγχουμε όλα τα κλειδιά των εγγραφών του block διαδοχικά. Όλα βασίζονται στην δομή που έχουμε για τα index και τα data blocks αντίστοιχα. Ανάλογα με τον operator και το κλειδί που μας δίνεται αλλά και το αντίστοιχο κλειδί της εγγραφής, αποφασίζουμε αν είναι κάποια από αυτές που ζητάμε. Τέλος, περνάμε στον πίνακα open_scans ένα object scan που έχει και τα results και γυρνάμε το scanDesc που είναι η θέση του global πίνακα που βάζουμε τις πληροφορίες του συγκεκριμένου scan.