Πληροφορική & Τηλεπικοινωνίες
Κ18 - Υλοποίηση Συστημάτων Βάσεων Δεδομένων
Εαρινό Εξάμηνο 2009 – 2010

Καθηγητής Δ. Γουνόπουλος

Άσκηση 2

Σε συνέχεια της πρώτης άσκησης, σκοπός της δεύτερης εργασίας είναι η καλύτερη κατανόηση των διαφορετικών τύπων κατακερματισμού που υπάρχουν. Πιο συγκεκριμένα, θα υλοποιήσετε ένα σύνολο συναρτήσεων που διαχειρίζονται αρχεία τα οποία δημιουργήθηκαν βάσει γραμμικού κατακερματισμού (Linear Hash). Επιπλέον, καλείστε να υλοποιήσετε διαφορετικές συναρτήσεις κατακερματισμού, και να αξιολογήσετε την απόδοσή τους σε δεδομένα τα οποία θα δίνονται, ώστε να γίνει κατανοητό το πόσο μπορεί να επηρεάσει την απόδοση ενός ΣΔΒΔ η συνάρτηση κατακερματισμού.

Οι συναρτήσεις που καλείστε να υλοποιήσετε αφορούν τη διαχείριση ευρετηρίων που στηρίζονται στο γραμμικό κατακερματισμό. Η υλοποίησή τους θα γίνει πάνω από το επίπεδο διαχείρισης μπλοκ *υποχρεωτικά*, το οποίο δίνεται έτοιμο ως βιβλιοθήκη. Τα πρωτότυπα (definitions) των συναρτήσεων που καλείστε να υλοποιήσετε όσο και των συναρτήσεων της βιβλιοθήκης επιπέδου μπλοκ δίνονται στη συνέχεια, μαζί με επεξήγηση για τη λειτουργικότητα που θα επιτελεί η κάθε μία.

Η διαχείριση των αρχείων γραμμικού κατακερματισμού γίνεται μέσω των συναρτήσεων με το πρόθεμα *LH_*. Τα αρχεία αυτά έχουν μέσα ***εγγραφές***, τις οποίες διαχειρίζεστε μέσω των κατάλληλων συναρτήσεων. Οι εγγραφές είναι της μορφής που δίνεται στη συνέχεια:

```
typedef struct{
        int id,
        char name[15],
        char surname[20],
        char city[10];
}Record;
```

Το πρώτο μπλοκ (block) κάθε αρχείου γραμμικού κατακερματισμού περιλαμβάνει *"ειδική"* πληροφορία (ή αλλιώς *μετα-πληροφορία*) σχετικά με το ίδιο το αρχείο. Το είδος της πληροφορίας αυτής εξαρτάται από την υλοποίηση του καθενός. Τέτοιου είδους πληροφορία αφορά π.χ. το πεδίο κατακερματισμού, τον τύπο του πεδίου κ.λπ.

Στη συνέχεια δίνεται ένα παράδειγμα των εγγραφών που θα προστίθενται στα αρχεία που θα δημιουργείτε με την υλοποίησή σας. Εγγραφές με τις οποίες μπορείτε να ελέγξετε το πρόγραμμά σας θα δοθούν υπό μορφή αρχείων κειμένου, μαζί με κατάλληλες συναρτήσεις main στο site του μαθήματος.

{15, "Giorgos", "Papadopoulos", "Ioannina"}
{4, "Tasos", "Politis", "Athina"}
{300, "Yannis", "Stratakis", "Larissa"}

Συναρτήσεις BF (Block File)

Στη συνέχεια, περιγράφονται οι συναρτήσεις που αφορούν το επίπεδο από block, πάνω στο οποίο θα βασιστείτε για την υλοποίηση των συναρτήσεων που ζητούνται. Η υλοποίηση των συναρτήσεων αυτών θα δοθεί έτοιμη με τη μορφή βιβλιοθήκης. Στο αρχείο κεφαλίδας **BF.h** που θα σας δοθεί υπάρχουν τα πρωτότυπα των συναρτήσεων μαζί με σχόλια για το πώς δουλεύουν και το μέγεθος ενός μπλοκ που είναι ***BF_BLOCK_SIZE***, ίσο με ***512*** bytes.

void BF_Init()

Η συνάρτηση *BF_Init* αρχικοποιεί το επίπεδο BF.

int BF_CreateFile(char filename /* όνομα αρχείου */)*

Η συνάρτηση *BF_CreateFile* δημιουργεί ένα αρχείο με όνομα *filename* το οποίο αποτελείται από block. Αν το αρχείο υπάρχει ήδη το παλιό αρχείο διαγράφεται. Σε περίπτωση επιτυχούς εκτέλεσης επιστρέφει 0, ενώ σε περίπτωση αποτυχίας επιστρέφει ένα αρνητικό αριθμό. Αν θέλετε να δείτε το είδος του λάθους μπορείτε να καλέσετε τη συνάρτηση *BF_PrintError*.

int BF_OpenFile(char filename /* όνομα αρχείου */)*

Η συνάρτηση *BF_OpenFile* ανοίγει ένα υπάρχον αρχείο από block με όνομα *filename*. Σε περίπτωση επιτυχίας επιστρέφει το αναγνωριστικό του αρχείου, ενώ σε περίπτωση αποτυχίας επιστρέφει ένα αρνητικό αριθμό. Αν θέλετε να δείτε το είδος του λάθους μπορείτε να καλέσετε τη συνάρτηση *BF_PrintError*.

int BF_CloseFile(int fileDesc / αναγνωριστικό αρχείου block */)*

Η συνάρτηση *BF_CloseFile* κλείνει το ανοιχτό αρχείο με αναγνωριστικό αριθμό *fileDesc*. Σε περίπτωση επιτυχούς εκτέλεσης επιστρέφει 0, ενώ σε περίπτωση αποτυχίας επιστρέφει ένα αρνητικό αριθμό. Αν θέλετε να δείτε το είδος του λάθους μπορείτε να καλέσετε τη συνάρτηση *BF_PrintError*.

int BF_GetBlockCounter(int fileDesc / αναγνωριστικό αρχείου block */)*

Η συνάρτηση *BF_GetBlockCounter* δέχεται ως όρισμα τον αναγνωριστικό αριθμό *fileDesc* ενός ανοιχτού αρχείου από block και επιστρέφει τον αριθμό των διαθέσιμων block του σε περίπτωση επιτυχούς εκτέλεσης. Σε περίπτωση αποτυχίας, επιστρέφει ένα αρνητικό αριθμό. Αν θέλετε να δείτε το είδος του λάθους μπορείτε να καλέσετε τη συνάρτηση *BF_PrintError*.

int BF_AllocateBlock(int fileDesc / αναγνωριστικό αρχείου block */)*

Με τη συνάρτηση *BF_AllocateBlock* δεσμεύεται ένα καινούριο block για το αρχείο με αναγνωριστικό αριθμό *fileDesc*. Το νέο block αρχικοποιείται με μηδενικά και δεσμεύεται πάντα στο τέλος του αρχείου, οπότε ο αριθμός του block είναι *BF_GetBlockCounter(fileDesc) - 1*. Σε περίπτωση επιτυχούς εκτέλεσης επιστρέφει 0, ενώ σε περίπτωση αποτυχίας επιστρέφει ένα αρνητικό αριθμό. Αν θέλετε να δείτε το είδος του λάθους μπορείτε να καλέσετε τη συνάρτηση *BF_PrintError*.

int BF_ReadBlock(
int fileDesc, / αναγνωριστικό αρχείου block */*
int blockNumber, / ο αριθμός ενός δεσμευμένου block μέσα στο αρχείο */*
*void** block /* δείκτης στα δεδομένα του block (αναφορά) */)*

Η συνάρτηση *BF_ReadBlock* βρίσκει το block με αριθμό *blockNumber* του ανοιχτού αρχείου με αναγνωριστικό αριθμό *fileDesc*. Η μεταβλητή *block* αποτελεί ένα δείκτη στα δεδομένα του block, το οποίο θα πρέπει να έχει δεσμευτεί. Σε περίπτωση επιτυχούς εκτέλεσης επιστρέφει 0, ενώ σε περίπτωση αποτυχίας επιστρέφει ένα αρνητικό αριθμό. Αν θέλετε να δείτε το είδος του λάθους μπορείτε να καλέσετε τη συνάρτηση *BF_PrintError*.

int BF_WriteBlock(
int fileDesc, / αναγνωριστικό αρχείου block */*
int blockNumber / ο αριθμός ενός δεσμευμένου block μέσα στο αρχείο */)*

Η συνάρτηση *BF_WriteBlock* γράφει στο δίσκο το δεσμευμένο block με αριθμό *blockNumber* του ανοιχτού αρχείου με αναγνωριστικό αριθμό *fileDesc*. Σε περίπτωση επιτυχούς εκτέλεσης επιστρέφει 0, ενώ σε περίπτωση αποτυχίας επιστρέφει ένα αρνητικό αριθμό. Αν θέλετε να δείτε το είδος του λάθους μπορείτε να καλέσετε τη συνάρτηση *BF_PrintError*.

void BF_PrintError(*char* message /* μήνυμα προς εκτύπωση */)*

Η συνάρτηση *BF_PrintError* βοηθά στην εκτύπωση των σφαλμάτων που δύναται να υπάρξουν με την κλήση συναρτήσεων του επιπέδου αρχείου block. Εκτυπώνεται στο stderr το *message* ακολουθούμενο από μια περιγραφή του πιο πρόσφατου σφάλματος.

Συναρτήσεις LH (Linear Hash)

Στη συνέχεια περιγράφονται οι συναρτήσεις που καλείστε να υλοποιήσετε και αφορούν το επίπεδο αρχείων γραμμικού κατακερματισμού.

int LH_CreateIndex(
*char *fileName, /* όνομα αρχείου */*
char attrName, /* το όνομα του πεδίου που είναι το κλειδί για το συγκεκριμένο αρχείο */*
char attrType, / τύπος πεδίου-κλειδιού: 'c', 'i' */*
int attrLength, / μήκος πεδίου-κλειδιού */*
int buckets, / αρχικό πλήθος κάδων κατακερματισμού */*
float loadThrs / κατώφλι του παράγοντα φόρτωσης αρχείου για διάσπαση */)*

Η συνάρτηση *LH_CreateIndex* χρησιμοποιείται για τη δημιουργία και κατάλληλη αρχικοποίηση ενός άδειου αρχείου γραμμικού κατακερματισμού με όνομα *fileName*. Σε περίπτωση επιτυχίας επιστρέφει 0, ενώ σε περίπτωση αποτυχίας επιστρέφει -1.

LH_info* LH_OpenIndex(*char *fileName /* όνομα αρχείου προς άνοιγμα */* ***)***

Η συνάρτηση *LH_OpenIndex* ανοίγει το αρχείο με όνομα *filename* και διαβάζει την μετα-πληροφορία του αρχείου γραμμικού κατακερματισμού. Κατόπιν, ενημερώνει μια δομή όπου κρατούνται όσες πληροφορίες κρίνετε απαραίτητες προκειμένου να μπορείτε να επεξεργαστείτε στη συνέχεια τις εγγραφές του. Μια ενδεικτική δομή με τέτοιες πληροφορίες δίνεται στη συνέχεια:

```
typedef struct {
        int fileDesc; /* αναγνωριστικός αριθμός ανοίγματος αρχείου από το επίπεδο block */
        char* attrName; /* το όνομα του πεδίου που είναι το κλειδί για το συγκεκριμένο αρχείο */
        char attrType,      /* τύπος πεδίου-κλειδιού: 'c', 'i' */
        int attrLength,     /* μήκος πεδίου-κλειδιού */
        int buckets,        /* αρχικό πλήθος κάδων κατακερματισμού */
        float loadThrs      /* κατώφλι του παράγοντα φόρτωσης αρχείου για διάσπαση */
} LH_info;
```

Όπου *attrName*, *attrType* και *attrLength* αφορούν το πεδίο-κλειδί, *fileDesc* είναι ο αναγνωριστικός αριθμός του ανοίγματος του αρχείου, όπως επιστράφηκε από το επίπεδο διαχείρισης μπλοκ και *numBuckets* είναι το πλήθος των κάδων που υπάρχουν στο αρχείο. Τέλος, *loadThrs* είναι ένα μέγιστο κατώφλι του επιτρεπόμενου παράγοντα φόρτωσης του αρχείου. Σε περίπτωση επιτυχίας επιστρέφει τη δομή πληροφοριών του αρχείου, ενώ σε περίπτωση αποτυχίας επιστρέφει τιμή NULL.

int LH_CloseIndex(*LH_info* header_info /* πληροφορίες του αρχείου προς κλείσιμο */* ***)***

Η συνάρτηση *LH_CloseIndex* κλείνει το αρχείο γραμμικού κατακερματισμού που προσδιορίζεται με τη δομή *header_info*. Η συνάρτηση είναι υπεύθυνη και για την αποδέσμευση της μνήμης που καταλαμβάνει η δομή που περάστηκε ως παράμετρος, στην περίπτωση που το κλείσιμο πραγματοποιήθηκε επιτυχώς. Σε περίπτωση επιτυχίας επιστρέφει 0, ενώ σε διαφορετική περίπτωση -1.

int LH_InsertEntry (
LH_info header_info, /* επικεφαλίδα του αρχείου*/*
Record record / δομή που προσδιορίζει την εγγραφή */* ***)***

Η συνάρτηση *LH_InsertEntry* εισάγει μία εγγραφή στο αρχείο γραμμικού κατακερματισμού, σύμφωνα και με την αντίστοιχη θεωρία. Οι πληροφορίες που αφορούν το αρχείο βρίσκονται στη δομή *header_info,* όπως αυτές επιστράφηκαν από την *LH_OpenIndex*, ενώ η εγγραφή προς εισαγωγή δίνεται στην παράμετρο *record*. Αν μετά την εισαγωγή της εγγραφής ο παράγοντας φόρτωσης του αρχείου είναι μεγαλύτερος από το σχετικό κατώφλι, τότε θα πρέπει να γίνεται διάσπαση των buckets. Ο παράγοντας φόρτωσης του αρχείου λαμβάνει υπόψη του μόνο το πλήθος των buckets και όχι τα blocks υπερχείλισης που μπορεί να υπάρχουν. Σε περίπτωση που εκτελεστεί επιτυχώς, επιστρέφει 0, ενώ σε διαφορετική περίπτωση -1.

int LH_FindAllEntries(
LH_info header_info, / επικεφαλίδα του αρχείου /*
*void *value, /* τιμή του πεδίου-κλειδιού προς αναζήτηση */* ***)***

Η συνάρτηση αυτή χρησιμοποιείται για την εκτύπωση όλων των εγγραφών που υπάρχουν στο αρχείο γραμμικού κατακερματισμού οι οποίες έχουν τιμή στο πεδίο-κλειδί ίση με *value*. Η παράμετρος *header_info* δίνει πληροφορίες για το αρχείο κατακερματισμού, όπως είχαν επιστραφεί από την *LH_OpenIndex*. Για κάθε εγγραφή

του αρχείου με τιμή στο πεδίο-κλειδί ίση με *value*, εκτυπώνονται τα περιεχόμενά της (συμπεριλαμβανομένου και του πεδίου-κλειδιού). Να εκτυπώνει επίσης το πόσα blocks διαβάστηκαν μέχρι να βρεθούν όλες οι εγγραφές. Σε περίπτωση επιτυχίας επιστρέφει το πλήθος των εγγραφών που τυπώθηκαν, ενώ σε περίπτωση λάθους επιστρέφει -1.

Συναρτήσεις Κατακερματισμού

Να υλοποιήσετε τουλάχιστον δύο (2) συναρτήσεις κατακερματισμού, και στα πλαίσια της εργασίας να τις αξιολογήσετε ως προς:

α) Το πόσα blocks έχει ένα αρχείο
β) Το ελάχιστο, το μέσο και το μέγιστο πλήθος εγγραφών που έχει κάθε bucket ενός αρχείου
γ) Το μέσο αριθμό των blocks που έχει κάθε bucket
δ) Το πλήθος των buckets που έχουν μπλοκ υπερχείλισης, και πόσα μπλοκ είναι αυτά για κάθε bucket.

Για το σκοπό αυτό, να υλοποιήσετε τη συνάρτηση:

int HashStatistics(*char* filename /* όνομα του αρχείου που ενδιαφέρει */* ***)***

Η συνάρτηση διαβάζει το αρχείο με όνομα *filename* και τυπώνει τα στατιστικά που αναφέρθηκαν προηγουμένως. Σε περίπτωση επιτυχίας επιστρέφει 0, ενώ σε περίπτωση λάθους επιστρέφει -1.

Παράδοση εργασίας

Η εργασία μπορεί να γίνει ατομικά ή ομαδικά (**2 ατόμων**).

Γλώσσα υλοποίησης: C / C++

Περιβάλλον υλοποίησης: Linux (gcc 4.3+) ή Windows (Visual Studio 2008). Η επιλογή έγινε με βάση τα διαθέσιμα εργαλεία ανάπτυξης που υπάρχουν στη σχολή.

Παραδοτέα: Τα αρχεία πηγαίου κώδικα (sources) και τα αντίστοιχα αρχεία κεφαλίδας (headers) καθώς και readme αρχείο με περιγραφή / σχόλια πάνω στην υλοποίησή σας.

Ημερομηνία Παράδοσης: 31 Μαΐου

Τρόπος παράδοσης: Ηλεκτρονικά, σύμφωνα και με τις οδηγίες που θα δοθούν στο site του μαθήματος (www.di.uoa.gr/~k18).