Πληροφορική & Τηλεπικοινωνίες
Υλοποίηση Συστημάτων Βάσεων Δεδομένων -
Χειμερινό Εξάμηνο 2016 – 2017
Καθηγητής Δ. Γουνόπουλος

Άσκηση 1 - Προθεσμία: 9 Δεκ. 2016

Σκοπός της εργασίας αυτής είναι η κατανόηση της εσωτερικής λειτουργίας των Συστημάτων Βάσεων Δεδομένων όσον αφορά τη διαχείριση σε επίπεδο μπλοκ (block) αλλά και ως προς τη διαχείριση σε επίπεδο εγγραφών. Επίσης, μέσω της εργασίας θα γίνει αντιληπτό το κατά πόσο μπορεί να βελτιώσει την απόδοση ενός Συστήματος Διαχείρησης Βάσεων Δεδομένων (ΣΔΒΔ) η ύπαρξη ευρετηρίων πάνω στις εγγραφές. Πιο συγκεκριμένα, στα πλαίσια της 1η εργασίας θα υλοποιήσετε ένα σύνολο συναρτήσεων που διαχειρίζονται αρχεία που δημιουργήθηκαν βάσει στατικού κατακερματισμού (Hash Table) και ένα σύνολο συναρτήσεων που διαχειρίζονται αρχεία που δημιουργήθηκαν βάσει επεκτατού κατακερματισμού (Extendible Hash Table).

Οι συναρτήσεις που καλείστε να υλοποιήσετε αφορούν τη διαχείριση εγγραφών και τη διαχείριση ευρετηρίων. 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 στατικού κατακερματισμού γίνεται μέσω των συναρτήσεων με το πρόθεμα HT_, ενώ των αρχείων επεκτατού κατακερματισμού με το πρόθεμα ET_. Τόσο τα αρχεία στατικού κατακερματισμού όσο και τα αρχεία επεκτατού κατακερματισμού έχουν μέσα εγγραφές τις οποίες διαχειρίζεστε μέσω των κατάλληλων συναρτήσεων. Οι **εγγραφές** είναι της ***ίδιας μορφής*** και για τους δύο τύπους αρχείων και συγκεκριμένα έχουν τη μορφή που δίνεται στη συνέχεια.

```
typedef struct{
        int id,
        char name[15],
        char surname[20],
        char city[25];
}Record;
```

Το πρώτο μπλοκ (block) κάθε αρχείου (είτε στατικού κατακερματισμού, είτε επεκτατού κατακερματισμού), περιλαμβάνει “ειδική” πληροφορία σχετικά με το ίδιο το αρχείο. Η πληροφορία αυτή χρησιμεύει στο να αναγνωρίσει κανείς αν πρόκειται για αρχείο στατικού ή επεκτατού κατακερματισμού, σε πληροφορία που αφορά το πεδίο κατακερματισμού για τα αντίστοιχα αρχεία 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 eclass του μαθήματος.

{15, “Giorgos”, “Dimopoulos”, “Ioannina”}
{4, “Antonia”, “Papadopoulou”, “Athina”}
{300, “Yannis”, “Yannakis”, “Thessaloniki”}

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()
Με τη συνάρτηση BF_Init πραγματοποιείται η αρχικοποίηση του επιπέδου 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 της συνάρτησης επιστρέφεται 0, ενώ σε περίπτωση αποτυχίας, επιστρέφεται ένας αρνητικός αριθμός. 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, επιστρέφεται το αναγνωριστικό του αρχείου, ενώ σε περίπτωση αποτυχίας, επιστρέφεται ένας αρνητικός αριθμός. 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 επιτυχίας επιστρέφει 0, ενώ σε περίπτωση αποτυχίας, επιστρέφεται ένας αρνητικός αριθμός. 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 Get_BlockCounter δέχεται ως όρισμα τον αναγνωριστικό αριθμό fileDesc ενός ανοιχτού αρχείου από block και βρίσκει τον αριθμό των διαθέσιμων block του, τον οποίο και επιστρέφει σε περίπτωση επιτυχούς εκτέλεσης. Σε περίπτωση αποτυχίας, επιστρέφεται ένας αρνητικός αριθμός. 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 επιτυχίας επιστρέφει 0, ενώ σε περίπτωση αποτυχίας, επιστρέφεται ένας αρνητικός αριθμός. 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 επιτυχίας, επιστρέφει 0, ενώ σε περίπτωση αποτυχίας, επιστρέφεται ένας αρνητικός αριθμός. 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 επιτυχίας, επιστρέφει 0, ενώ σε περίπτωση αποτυχίας, επιστρέφεται ένας αρνητικός αριθμός. 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 HT (Hash Table)

Στη συνέχεια περιγράφονται οι συναρτήσεις που καλείστε να υλοποιήσετε στα πλαίσια της εργασίας αυτής και που αφορούν το αρχείο ***στατικού κατακερματισμού***.

int HT_CreateIndex(char *fileName, /* όνομα αρχείου */
char attrType, /* τύπος πεδίου-κλειδιού: 'c', 'i' */
char* attrName, /* όνομα πεδίου-κλειδιού */
int attrLength, /* μήκος πεδίου-κλειδιού */
int buckets /* αριθμός κάδων κατακερματισμού*/
)

Η συνάρτηση HT_CreateIndex χρησιμοποιείται για τη δημιουργία και κατάλληλη αρχικοποίηση ενός άδειου αρχείου κατακερματισμού με όνομα fileName. Σε περίπτωση που εκτελεστεί επιτυχώς, επιστρέφεται 0, ενώ σε διαφορετική περίπτωση -1.

HT_info* HT_OpenIndex(char *fileName /* όνομα αρχείου */)
Η συνάρτηση HT_OpenIndex ανοίγει το αρχείο με όνομα filename και διαβάζει από το πρώτο μπλοκ την πληροφορία που αφορά το αρχείο κατακερματισμού. Κατόπιν, ενημερώνετε μια δομή όπου κρατάτε όσες πληροφορίες κρίνετε αναγκαίες για το αρχείο αυτό προκειμένου να μπορείτε να επεξεργαστείτε στη συνέχεια τις εγγραφές του. Μια ενδεικτική δομή με τις πληροφορίες που πρέπει να κρατάτε δίνεται στη συνέχεια:

```
typedef struct {
        int fileDesc;           /* αναγνωριστικός αριθμός ανοίγματος αρχείου από το επίπεδο block */
        char attrType;    /* ο τύπος του πεδίου που είναι κλειδί για το συγκεκριμένο αρχείο, 'c' ή'i' */
        char* attrName;         /* το όνομα του πεδίου που είναι κλειδί για το συγκεκριμένο αρχείο */
        int attrLength;         /* το μέγεθος του πεδίου που είναι κλειδί για το συγκεκριμένο αρχείο */
        long int numBuckets;            /* το πλήθος των "κάδων" του αρχείου κατακερματισμού */
} HT_info;
```

Όπου *attrType*, *attrName*, 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. Αφού ενημερωθεί κατάλληλα η δομή πληροφοριών του αρχείου, την επιστρέφετε.
Σε περίπτωση που συμβεί οποιοδήποτε σφάλμα, επιστρέφεται τιμή NULL. Αν το αρχείο που δόθηκε για άνοιγμα δεν αφορά αρχείο κατακερματισμού, τότε αυτό επίσης θεωρείται σφάλμα.

int HT_CloseIndex(HT_info* header_info)
Η συνάρτηση HT_CloseIndex κλείνει το αρχείο που προσδιορίζεται μέσα στη δομή header_info. Σε περίπτωση που εκτελεστεί επιτυχώς, επιστρέφεται 0, ενώ σε διαφορετική περίπτωση -1. 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.

int HT_InsertEntry(HT_info header_info, /* επικεφαλίδα του αρχείου*/
Record record /* δομή που προσδιορίζει την εγγραφή */)
Η συνάρτηση HT_InsertEntry χρησιμοποιείται για την εισαγωγή μίας εγγραφής στο αρχείο κατακερματισμού. Οι πληροφορίες που αφορούν το αρχείο βρίσκονται στη δομή header_info, ενώ η εγγραφή προς εισαγωγή προσδιορίζεται από τη δομή record. Σε περίπτωση που εκτελεστεί επιτυχώς, επιστρέφεται 0, ενώ σε διαφορετική περίπτωση -1.

int HT_GetAllEntries(HT_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 κατακερματισμού οι οποίες έχουν τιμή στο πεδίο-κλειδί ίση με value. Η πρώτη δομή δίνει πληροφορία

για το αρχείο κατακερματισμού, όπως αυτή είχε επιστραφεί από την HT_OpenIndex. Για κάθε εγγραφή που υπάρχει στο αρχείο και έχει τιμή στο πεδίο-κλειδί (όπως αυτό ορίζεται στην HT_info) ίση με value, εκτυπώνονται τα περιεχόμενά της (συμπεριλαμβανομένου και του πεδίου-κλειδιού). Να επιστρέφεται επίσης το πλήθος των blocks που διαβάστηκαν μέχρι να βρεθούν όλες οι εγγραφές. Σε περίπτωση επιτυχίας επιστρέφει το πλήθος των εγγραφών που τυπώθηκαν, ενώ σε περίπτωση λάθους επιστρέφει -1.

Συναρτήσεις ET (Extendible Hash Table)

Στην συνέχεια περιγράφονται οι συναρτήσεις που καλείστε να υλοποιήσετε και αφορούν στο επίπεδο αρχείων ***επεκτατού κατακερματισμού***.

int EH_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 depth                    /* το ολικό βάθος ευρετηρίου επεκτατού κατακερματισμού */
)
```
Η συνάρτηση EH_CreateIndex χρησιμοποιείται για τη δημιουργία και την κατάλληλη αρχικοποίηση ενός άδειου αρχείου επεκτατού κατακερματισμού με όνομα fileName. Σε περίπτωση επιτυχίας επιστρέφει 0, ενώ σε περίπτωση αποτυχίας επιστρέφει -1.

EH_info* EH_OpenIndex(char *fileName /* όνομα αρχείου προς άνοιγμα */)

Η συνάρτηση EH_OpenIndex ανοίγει το αρχείο με όνομα filename και διαβάζει την μετα-πληροφορία του αρχείου επεκτατού 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 depth                    /* το ολικό βάθος ευρετηρίου επεκτατού κατακερματισμού */
} EH_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 *depth* είναι το ολικό βάθος του ευρετηρίου επεκτατού κατακερματισμού. Σε περίπτωση επιτυχίας επιστρέφει τη δομή πληροφοριών του αρχείου, ενώ σε περίπτωση αποτυχίας επιστρέφει τιμή NULL. Αν το αρχείο που δοθηκε για άνοιγμα δεν αφορά αρχείο επεκτατού κατακερματισμού, τότε αυτό επίσης θεωρείται σφάλμα.

int EH_CloseIndex(EH_info* header_info /* πληροφορίες του αρχείου προς κλείσιμο */)
Η συνάρτηση EH_CloseIndex κλείνει το αρχείο επεκτατού 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 EH_InsertEntry(EH_info* header_info, /* επικεφαλίδα του αρχείου*/
Record record /* δομή που προσδιορίζει την εγγραφή */)
Η συνάρτηση EH_InsertEntry εισάγει μία εγγραφή στο αρχείο επεκτατού κατακερματισμού, σύμφωνα και με την αντίστοιχη θεωρία. Οι πληροφορίες που αφορούν στο αρχείο βρίσκονται στη δομή header_info, όπως αυτές επιστράφηκαν από την EH_OpenIndex, ενώ η εγγραφή προς εισαγωγή δίνεται στην παράμετρο *record*. Σε περίπτωση που εκτελεστεί επιτυχώς, επιστρέφει 0, ενώ σε διαφορετική περίπτωση -1.

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

Αφου υλοποιήσετε τις συναρτήσεις κατακερματισμού, 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, και
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 είναι ομαδική **2 ατόμων**.
Ημερομηνία Παράδοσης: 9 Δεκ. 2016
Γλώσσα υλοποίησης: C / C++

Παραδοτέα: Τα αρχεία πηγαίου κώδικα (sources) και τα αντίστοιχα αρχεία κεφαλίδας (headers), readme αρχείο με περιγραφή / σχόλια πάνω στην υλοποίησή σας. Καθώς και τις αντίστοιχες main συναρτήσεις με τις οποίες δοκιμάσατε την λειτουργικότητα του κώδικα σας.
Για απορίες, κάντε signup εδώ: piazza.com/uoa.gr/fall2016/18