Πληροφορική & Τηλεπικοινωνίες

Κ18 - Υλοποίηση Συστημάτων Βάσεων Δεδομένων

Εαρινό Εξάμηνο 2009 – 2010

Καθηγητής Δ. Γουνόπουλος

Άσκηση 1

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 ΣΔΒΔ η ύπαρξη ευρετηρίων πάνω στις εγγραφές. Πιο συγκεκριμένα, στα πλαίσια της 1η εργασίας θα υλοποιήσετε ένα σύνολο συναρτήσεων που διαχειρίζονται αρχεία σωρού (Heap Files) και ένα σύνολο συναρτήσεων που διαχειρίζονται αρχεία που δημιουργήθηκαν βάσει στατικού κατακερματισμού (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 σωρού γίνεται μέσω των συναρτήσεων με το πρόθεμα HP_, ενώ των αρχείων στατικού κατακερματισμού με το πρόθεμα HT_. Τόσο τα αρχεία σωρού όσο και τα αρχεία στατικού 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[10];
}Record;
```

Το πρώτο μπλοκ (block) κάθε αρχείου (είτε σωρού είτε κατακερματισμού), περιλαμβάνει *"ειδική"* πληροφορία σχετικά με το ίδιο το αρχείο. Η πληροφορία αυτή χρησιμεύει στο να αναγνωρίσει κανείς αν πρόκειται για αρχείο σωρού ή για αρχείο 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 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()

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 HP (Heap File)

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

int HP_CreateFile(*char *fileName, /* όνομα αρχείου */)*

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

int HP_OpenFile(*char *fileName /* όνομα αρχείου */* ***)***

Η συνάρτηση HP_OpenFile ανοίγει το αρχείο με όνομα *filename* και διαβάζει από το πρώτο μπλοκ την πληροφορία που αφορά το αρχείο σωρού. Επιστρέφει τον αναγνωριστικό αριθμό ανοίγματος αρχείου, όπως αυτός επιστράφηκε από το επίπεδο διαχείρισης μπλοκ ή -1 σε περίπτωση σφάλματος. Αν το αρχείο που ανοίχτηκε δεν πρόκειται για αρχείο σωρού, τότε αυτό θεωρείται επίσης περίπτωση σφάλματος.

int HP_CloseFile(*int fileDesc /* αναγνωριστικός αριθμός ανοίγματος αρχείου */* ***)***

Η συνάρτηση *HP_CloseFile* κλείνει το αρχείο που προσδιορίζεται από τον αναγνωριστικό αριθμό ανοίγματος *fileDesc*. Σε περίπτωση που εκτελεστεί επιτυχώς, επιστρέφεται 0, ενώ σε διαφορετική περίπτωση -1.

int HP_InsertEntry (
int fileDesc, / αναγνωριστικός αριθμός ανοίγματος αρχείου */*
Record record / δομή που προσδιορίζει την εγγραφή */* ***)***

Η συνάρτηση *HP_InsertEntry* χρησιμοποιείται για την εισαγωγή μίας εγγραφής στο αρχείο σωρού. Ο αναγνωριστικός αριθμός ανοίγματος του αρχείου δίνεται με την *fileDesc* ενώ η εγγραφή προς εισαγωγή προσδιορίζεται από τη δομή *record*. Η εγγραφή προστίθεται στο τέλος του αρχείου, μετά την τρέχουσα τελευταία εγγραφή. Σε περίπτωση που εκτελεστεί επιτυχώς, επιστρέφεται 0, ενώ σε διαφορετική περίπτωση -1.

void HP_GetAllEntries(
int fileDesc, / αναγνωριστικός αριθμός ανοίγματος αρχείου */*
char fieldName, /* όνομα του πεδίου για το οποίο γίνεται ο έλεγχος */*
*void *value, /* τιμή του πεδίου προς σύγκριση */* ***)***

Η συνάρτηση αυτή χρησιμοποιείται για την εκτύπωση όλων των εγγραφών που υπάρχουν στο αρχείο σωρού οι οποίες έχουν τιμή στο πεδίο με όνομα *fieldName* ίση με *value*. Το *fileDesc* είναι ο αναγνωριστικός αριθμός ανοίγματος του αρχείου, όπως αυτός έχει επιστραφεί από το επίπεδο διαχείρισης μπλοκ. Η παράμετρος *fieldName* μπορεί να παίρνει μία από τις εξής τιμές: "***id***", "***name***", "***surname***" ή "***city***", αναφερόμενη στα αντίστοιχα πεδία μιας εγγραφής.
Για κάθε εγγραφή που βρίσκεται μέσα στο αρχείο και έχει τιμή *value* στο πεδίο με όνομα *fieldName*, εκτυπώνονται τα περιεχόμενά της (συμπεριλαμβανομένου και του πεδίου-κλειδιού). Να εκτυπώνεται επίσης το πλήθος των μπλοκ που διαβάστηκαν.

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. Το μέγιστο πλήθος κάδων που μπορεί να δημιουργηθούν για ένα αρχείο κατακερματισμού είναι **100**.

int HT_CreateIndex(
*char *fileName, /* όνομα αρχείου */*
char attrType, / τύπος πεδίου-κλειδιού: 'c', 'i', 'f' */*
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 το κλειδί για το συγκεκριμένο αρχείο, 'c' ή 'i' */
        char* attrName; /* το όνομα του πεδίου που είναι το κλειδί για το συγκεκριμένο αρχείο */
        int attrLength; /* το μέγεθος του πεδίου που είναι το 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.

void 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 εκτυπώνεται επίσης το πλήθος των μπλοκ που διαβάστηκαν.

Παράδοση εργασίας

Η εργασία είναι ομαδική **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: 30 Απριλίου