Υλοποίηση συστημάτων βάσεων δεδομένων

Παναγιώτης Μαντάς: 1115201400294

Μάριος Χριστοδούλου: 1115201800225

Άσκηση 1

Προσοχή ότι η HT_GetAllEntries έχει τροποποιηθεί ώστε να παίρνει int αντί void *.
Δηλαδή ακολουθήσαμε την εκφώνηση και όχι το interface του κώδικα.

Έχουν υλοποιηθεί όλες οι λειτουργίες της εκφώνησης.

Οι επικεφαλίδες είναι ως εξής:

	HP	**HT**

Δημόσια επικεφαλίδα	typedef struct {
	  int fd;
	  int records;
	  int density;
	} HP_info;
		typedef struct {
		  int fd;
		  int records;
		  int density;
		  int buckets;
		} HT_info;

Ουρά μπλοκ	typedef struct {
	  int records;
	  int next_block;
	} HP_block_info;
		typedef struct {
		  int records;
		  int next_block;
		} HT_block_info;

Επικεφαλίδα υλοποίησης	union Header {
	  struct {
	  char prefix[2];
	  HP_info info;
	  };
	  char block[BF_BLOCK_SIZE];
	};
		union Header {
		  struct {
		  char prefix[2];
		  HT_info info;
		  int head[N];
		  };
		  char block[BF_BLOCK_SIZE];
		};
		
		με N:
		
		N =(BF_BLOCK_SIZE-2- sizeof (HT_info)) / 4

Magic word	HP	HT

Ως hash function έχει χρησιμοποιηθεί η εξής (από ίντερνετ):

```
static unsigned int hash(unsigned int x) {
   x = ((x >> 16) ^ x) * 0x45d9f3b;
   x = ((x >> 16) ^ x) * 0x45d9f3b;
   x = (x >> 16) ^ x;
   return x;
}
```

Η main σε κάθε υλοποίηση εισάγει 1000 εγγραφές και κάνει αναζήτηση για κάθε μία από αυτές.

Ενδεικτική έξοδος για το στατικό κατακερματισμό (1000 εγγραφές):

Bucket	Blocks	Records	Min/block	Max/block	Avg/block	Overflow
0	17	100	4	6	5.88	true
1	13	74	2	6	5.69	true
2	17	99	3	6	5.82	true
3	20	118	4	6	5.90	true
4	18	103	1	6	5.72	true
5	16	91	1	6	5.69	true
6	20	115	1	6	5.75	true
7	17	100	4	6	5.88	true
8	17	100	4	6	5.88	true
9	17	100	4	6	5.88	true

```
Total file blocks          : 173
Min records per bucket     : 74
Max records per bucket     : 118
Avg records per bucket     : 100.00
Avg blocks  per bucket     : 17.20
Total buckets with overflow: 10
```

Ενδεικτική έξοδος για το στατικό κατακερματισμό (10000 εγγραφές):

Bucket	Blocks	Records	Min/block	Max/block	Avg/block	Overflow
0	174	1041	3	6	5.98	true
1	158	945	3	6	5.98	true
2	157	938	2	6	5.97	true
3	176	1055	5	6	5.99	true
4	169	1009	1	6	5.97	true
5	160	958	4	6	5.99	true
6	173	1035	3	6	5.98	true
7	171	1021	1	6	5.97	true
8	168	1004	2	6	5.98	true
9	166	994	4	6	5.99	true

```
Total file blocks          : 1673
Min records per bucket     : 938
Max records per bucket     : 1055
Avg records per bucket     : 1000.00
Avg blocks  per bucket     : 167.20
Total buckets with overflow: 10
```

Σύντομη περιγραφή δομής:

Στην επικεφαλίδα έχουμε για κάθε bucket τον αριθμό του πρώτου block number της αλυσίδας. Αν μία αλυσίδα είναι κενή δεν πιάνει χώρο καθόλου στο δίσκό.

Κάθε μπλοκ έχει ως μεταπληροφορίες πλήθος εγγραφών και αριθμό του επόμενου μπλοκ.

Για εισαγωγή:

1. Υπολογίζουμε τη τιμή της συνάρτησης κατακερματισμού για το συγκεκριμένο κλειδί
2. Βρίσκουμε από το μπλοκ 0 τον αριθμό του πρώτου block number της αλυσίδας
3. Και (αν υπάρχει) πάμε σε αυτό το μπλοκ.
4. Αν υπάρχει χώρος εκεί εισάγουμε και τέλος, Διαφορετικά επαναλαμβάνουμε από το βήμα 3 για το επόμενο μπλοκ.

5. Αν δεν υπάρχει τότε εισάγουμε νέο μπλοκ και ανανεώνουμε αντίστοιχα τους αριθμούς.

Άσκηση 2

Έχει υλοποιηθεί το δευτερεύων ευρετήριο ως εξής:

Ανάλογα με το record_attribute κάνουμε hashing στο αντίστοιχο πεδίο (record, name, surname, city) όπως στην άσκηση 1 με hash function (με τη διαφορά ότι εδώ έχουμε είσοδο συμβολοσειρά αντί αριθμό):

```
unsigned int hash(char *str) {
    unsigned long hash = 5381;
    int c;

    while ((c = *str++) != 0) {
        hash = ((hash << 5) + hash) + c; /* hash * 33 + c */
    }

    return hash;
}
```

Βάσει αυτής της hash function βρίσκουμε σε ποιο bucket θα μπει η εγγραφή και την τοποθετούμε εκεί. Αλγοριθμικά η διαδικασία είναι η ίδια με το κλασικό πίνακα κατακερματισμού και επίσης ότι ως εγγραφή αποθηκεύουμε το ζεύγος (record_attribute, block number).

Οι επικεφαλίδες είναι ως εξής:

ΣΗΤ

Δημόσια επικεφαλίδα

```
typedef struct {
    int fd;
    int records;
    int density;
    int buckets;
    char record_attribute[15];
    char primary_data_file[20];
} SHT_info;
```

Ουρά μπλοκ

```
typedef struct {
    int records;
    int next_block;
} SHT_block_info;
```

Επικεφαλίδα υλοποίησης

```
union Header {
    struct {
        char prefix[4];
        SHT_info info;
        int head[(BF_BLOCK_SIZE - 4 - sizeof (SHT_info)) / 4];
    };
    char block[BF_BLOCK_SIZE];
};
```

με N:

N =(BF_BLOCK_SIZE- 4 - sizeof (SHT_info)) / 4

Magic word SHT

Η main εκτελεί τις οδηγίες της εκφώνησης με 1000 εγγραφές και κάνει αναζήτηση για κάθε μία από αυτές βάσει και του primary και του secondary key.

Ενδεικτική εκτέλεση (1 εκτέλεση):

Για το HT:

```
HT File closed, HT_ERRORS: 0
Bucket       Blocks      Records    Min/block    Max/block    Avg/block     Overflow
    0           20          120            6            6         6.00         true
    1           18          106            4            6         5.89         true
    2           16           93            3            6         5.81         true
    3           17           99            3            6         5.82         true
    4           16           91            1            6         5.69         true
    5           17           99            3            6         5.82         true
    6           17          100            4            6         5.88         true
    7           16           95            5            6         5.94         true
    8           17          102            6            6         6.00         true
    9           16           95            5            6         5.94         true
Total file blocks          : 171
Min records per bucket     : 91
Max records per bucket     : 120
Avg records per bucket     : 100.00
Avg blocks  per bucket     : 17.00
Total buckets with overflow: 10
```

Για το SHT:

```
HT File closed, HT_ERRORS: 0
Bucket       Blocks      Records    Min/block    Max/block    Avg/block     Overflow
    0           15           85            1            6         5.67         true
    1           27          162            6            6         6.00         true
    2           32          187            1            6         5.84         true
    3           27          157            1            6         5.81         true
    4            0            0   2147483647            0         -nan        false
    5           16           91            1            6         5.69         true
    6           24          142            4            6         5.92         true
    7            0            0   2147483647            0         -nan        false
    8           17          101            5            6         5.94         true
    9           13           75            3            6         5.77         true
Total file blocks          : 172
Min records per bucket     : 0
Max records per bucket     : 187
Avg records per bucket     : 100.00
Avg blocks  per bucket     : 17.10
Total buckets with overflow: 8
```

Σύντομη περιγραφή δομής:

Η άσκηση 2 είναι αλγοριθμικά ίδια με τη προηγούμενη αλλά αντί για πλήρεις εγγραφές βάζουμε μόνο ζεύγη (key, blocknum). Εισαγωγή γίνεται ακριβώς με τον ίδιο τρόπο.

Για την αναζήτηση για κάθε τέτοιο ζεύγος πάμε στο αρχικό αρχείο και κάνουμε διάσχιση μόνο στο αντίστοιχο μπλοκ.