CS411 Database Systems

Fall 2009

HW#3

Due: 3:15pm CST, November 3, 2009

Note: Print your name and NetID in the upper right corner of every page of your submission. Hand in your stapled homework to Donna Coleman in 2106 SC. In case Donna is not in office, slide your homework under the door.

To grade homeworks faster, the homework is partitioned into two parts as follows:

- Part 1: Problem 1 - Problem 3
- Part 2: Problem 4 - Problem 5

Please, submit each part separately. For each part, make sure to write down your name and NetID.

Handwritten submissions will be graded but they will take longer to grade. For clarity, machine formatted text is preferable: Expect to lose points if your handwritten answer is unclear or misread by the grader.

Discussions with other students are strongly encouraged. However, such collaboration should be limited to general strategies and ideas, but not write-ups of specific solutions. Your submitted work should represent your identifiable efforts and originality.

Part 1

Problem 1 Representing Block and Record Addresses (20 points)

Suppose that we have 4096-byte blocks in which we store records of 100 bytes. The block header consists of an offset table, as in the figure below, using 2-byte pointers to records within the block.

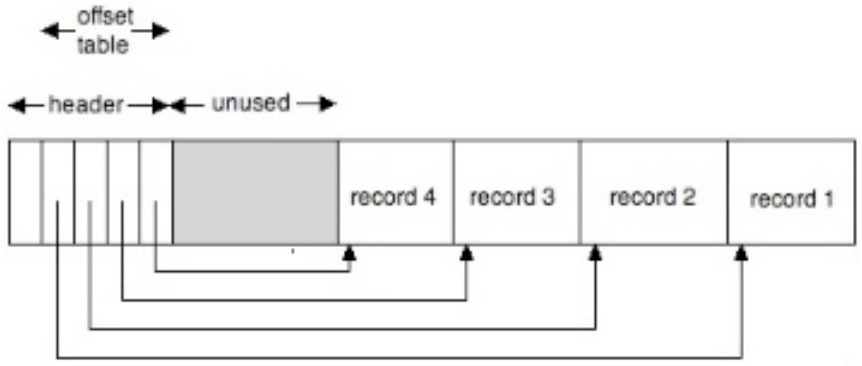

Figure 1: Block Header

a) What kind of information is stored in the block header in Figure 1 besides the offset table? (3 points)

b) Explain why we prefer to have unused area in the middle of a block when the block header contains an offset table? (3 points)

c) On an average day, two records per block are inserted, and one record is deleted. A deleted record must have its pointer replaced by a "tombstone", because there may be dangling pointers to it. For specificity, assume the deletion on any day always occurs before the insertions. If the block is initially empty, after how many days will there be no room to insert any more records? (10 points)

d) Redesign the layout of a block to store records more efficiently considering the fact the block stores *fixed-length* records. (4 points)

Problem 2 Spanned Records (15 points)

Suppose blocks have 1000 bytes available for the storage of records, and we wish to store on them fixed-length records of length r, where $500 < r \leq 1000$. The value of r includes the record header, but a record fragment requires an additional 16 bytes for the fragment header. For what values of r can we improve space utilization by spanning records?

Problem 3 Index structure basics (20 points)

Consider an index on a sequential file consisting of 10,000 blocks. Each block contains 10 fixed sized records. Each key value found in the file is unique. For this problem, assume that:

- Pointers to blocks are 10 bytes long.
- Pointers to records are 20 bytes long.
- Index blocks are 5000 bytes (in addition to the header).
- Search keys for file records are 10 bytes long.
- There is no buffered block in memory in the beginning, but it is possible to locate each block in the sparse index file.

(a) How many blocks do we need to hold a sparse one-level, primary index? (5 points)

(b) In (a), how many disk I/Os do we need to find and retrieve a record with a given key at the worst case? (5 points)

(c) How many blocks do we need to hold a one-level, secondary index? (5 points)

(d) Suppose that we introduce an additional second level index on the sparse index in (a). How many disk I/Os do we need to find and retrieve a record with a given key at the worst case? (5 points)

Part 2

Problem 4 B-Tree (25 points)

Consider a B-tree of degree $d = 2$, shown in Figure 2. Remember that each block has space for $2d$ keys and $2d + 1$ pointers. The textbook uses a different parameter n in Chapter 14.2.1, which is equal to $2d$. Please consider the execution of each operation in the following questions.

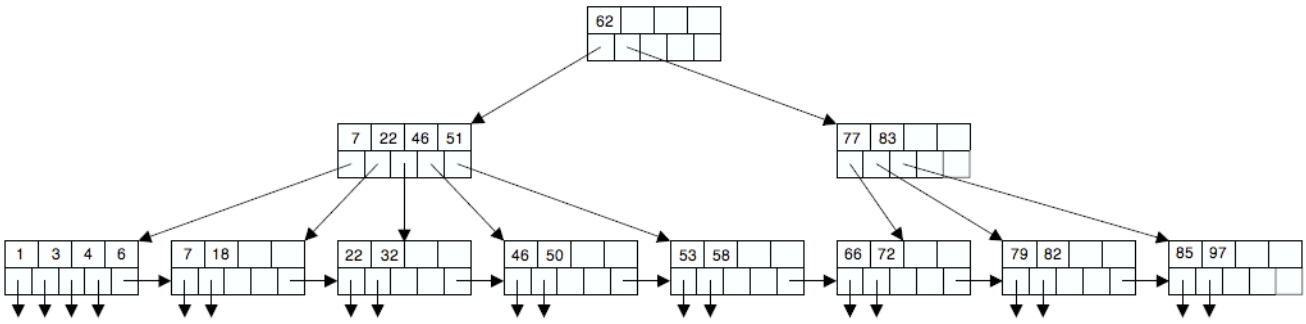

Figure 2: B tree

(a) Look up the record with key 82. Please describe how to traverse the tree in detail. (5 points)

(b) Look up the records with key in the range of 51 and 80. Please describe how to traverse the tree in detail. (6 points)

(c) Show the B-tree that would result from inserting a data entry with key 5 into the original tree. (7 points)

(d) Show the B-tree that would result from deleting the data entry with key 72 from the original tree. (7 points)

Problem 5 Hash Table (20 points)

Consider indexing the following key values using an extensible hash table. Keys are inserted in the following order:
34, 60, 51, 73, 49, 84, 25

The hash function h(n) for key n is h(n) = n mod 16; that is, the hash function is the reminder after the key value is divided by 16, giving the hash a 4-bit value. Assume that each bucket can hold 2 data items.

(a) Draw a hash table, which contains both the array of pointers in main memory and the buckets (i.e., data blocks) in secondary storage, after the first **four** keys are inserted. Show the keys along with their hash values in the buckets. Be sure to indicate the number of bits in the hash value that are used in the array (variable i). Also, indicate the "nub" value of each block. (10 points)

(b) Suppose that we use a linear hash table instead. Draw a hash table in the similar way, after the first **five** keys are inserted. You do not have to specify the "nub" value of each block in this question. Note that an extension of the table is necessary when the average number of records per block exceeds 80% of the number of records that fill one block. (10 points)