Assignment 6

Payung, Riley
Computational & Data Sciences, B.S.
George Mason University
Fairfax, VA, United States
rpayung@gmu.edu

Abstract

The following paper contains answers for the attributes of a B⁺ Tree and a Hash-File Organization of relations and databases. Here we will analyze the maximums, minimums, and best and worst-case scenarios of using both methods of storing a relation *R*. Both methods have their benefits and downfalls, however, that is partially out of the scope of this paper.

1. B⁺ Tree Index

Consider a relation R with 10,000 records stored as a file on the disk. We wish to build a B+-tree index using a candidate key of R as the search key. The size of the key value is 16 bytes, while the size of a pointer is 4 bytes.

Since the B+-tree is an index, the leaves of the tree contain pointers to the records of the data file of R (not the actual records). Each block (i.e., page) is 512 bytes.

1.1 Answer

The maximum number of total nodes and pointers that each node may have is 10000. The maximum number of values that this each node may have is up to 10000 since the maximum number of pointers that any node is n, and the maximum number of values that any intermediate node may have is between half of all nodes and n.

1.2 Answer

The minimum number of non-root pointers in this tree allow for 5000 nodes and pointers, since the minimum number of nodes that each one must be at least half.

1.3 Answer

The minimum number of blocks that this tree could have is 400 record sets. Since there's a total of 512 bytes in each block, we can fit 25 records (with some change, actual amount is 25.6, but you can't store half of a search-key record) in each. Now, the simple math of 10000 records (assuming there's search keys for all of these) divided by 25 records for each, gives us 400. 10000 / 25 = 400.

1.4 Answer

Let's take our answer of getting 25.6 records for each of our blocks, and divide it by 2, since we take n/2 for intermediate nodes. 25.6 / 2 = 12.8 (Lets round up to 13). 10000 / 13 = 769.23 → 770 total nodes is our worst case.

1.Bonus Answer

The cost of accessing in block reads for a record with the best case is given by

$$log_{25}(400) = 1.861353$$

block reads since we have 25 records per block and 400 blocks total.

The cost of accessing in block reads for a record with the worst case is

$$log_{13}(770) = 2.59124$$

block reads since we have 13 records per block and 770 blocks total.

2. Hash File Organization

Consider a relation R with 10,000 records. This relation has 4 attributes. The sizes of each attribute are: 6 bytes, 12 bytes, 4 bytes, and 18 bytes, respectively. We wish to store R as a hash file on the disk with 1,000 buckets. The size of a pointer (to an overflow block) is 4 bytes. It must be reserved in each bucket block. Each block is 4,096 bytes. Note that records cannot be split between different blocks

2.1 Answer

Each bucket should have at minimum 10 records since we want to store 10,000 records in a total of 1,000 buckets. 10,000 / 1,000 = 10.

2.2 Answer

The minimum number of records required to create an overflow block would be caused by the total amount of records being stored being greater than our block-reserved space, which is 4,096 bytes, or 4KB. Since each record is approx. 44 bytes in size, including our pointer, we would need to store at minimum 93 records to create a bucket overflow (4096 / 44 = 93.09). This would require that we create a new bucket linking it to the previous bucket. Luckily, since we only require 44 bytes, there is some space for a pointer to the next bucket in our previous bucket. 44 * 93 = 4092 + 4 bytes for a pointer equals 4096 exactly, using the maximum amount of space without any lost allocated space.

2.3 Answer

The total minimum number of buckets to store R would require 108 buckets (107.422 buckets). This number can be acquired by the given equation:

$$buckets = n * \frac{r + p}{bucket\_size}$$

Equation 1: Minimum Buckets for storage of items

Where *n* is the total number of records in the relation, *r* is the size of each record, *p* is our pointer size, and *bucket_size* is the size of a single bucket in the relation.

$$10000 * \frac{40 + 4}{4096} = 107.422$$

10,000 (number of records to be stored) * 44 (bytes per record) / 4096 (size of bucket)

2.4 Answer

The total maximum number of buckets to store R would be our upper limit in which we do use all 1000 buckets.

2.Bonus Answer

Our best case would be assuming that every record is stored and fills exactly 108 buckets. In that case, the best case would be given by:

$$O(1 + n)$$

Where n is the number of records in a single bucket, the 1 is from the fact that searching for a record in a hash is O(1) since we can just pass the search to our hash function and look-up the bucket using that value obtained.

Our worst case would be assuming that we used all 1000 buckets, or we had to chain more than one bucket together. If we had to chain more than 1 bucket, our worst case would be

$$O(1 + n^p)$$

Where n is the number of records inside of a single bucket and *p* is the number of chained buckets attached to our primary bucket.