Homework 2

Spring 2020

Due: 11:59PM EST, March 23, 2020. Submit using Blackboard.

(There will be a 10% penalty for each late day. After 5 late days, the homework will not be accepted.)

Part A: Disk and Files (3+3+4 = 10 pts)

1. Why variable density disk is desirable over equal density disk? Explain.
2. What is sequential flooding of the buffer pool?
3. Which page replacement policy (MRU or LRU) is better for the Selection operator? For the nested loops join operator? Why?

Part B: Tree-Based Indexing (50 pts)

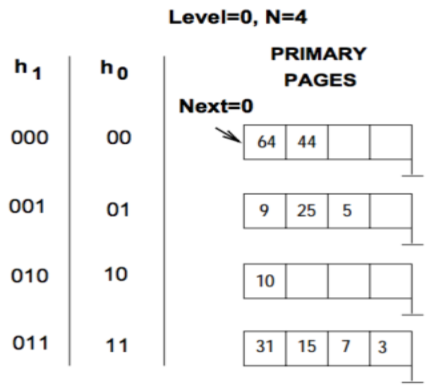
1. Show the result of inserting 10, 5, 20, 30, 45, 50, 12, 15, 16, 25, 32, 2 , 1, 23, 49 , 50 and 3 (in this order) into an initially empty B+-tree of order d = 2. Show every step (in drawing) after inserting each of the values. ((first 4x1pt) + (next 13x2pts) = 30 pts)
2. Show the result of deleting 50, 25, 3, 20 (in this order)) from the previous B+-tree. Show the B+-tree after each deletion. ((first 3x2pts) + (next 1*4pts =10 pts)
3. Assume that you have a database system where the workload is very dynamic (lots of inserts, updates and deletes). Now, you are asked to choose between the following index structures: ISAM and B+Tree. What will be your preferred index structure and why? Explain in detail by justifying your choice. Now, repeat Question B.3. if the database is entirely static. (10 pts)

Part C: Hash-based Indexing (40 pts)

1. Consider an extendible hashing structure where:

   a. Initially each bucket can hold up to 3 records.
   b. The directory is an array of Size 4. Now, construct the extendible hashing structure by inserting the keys 2, 3, 7, 14, 15, 18, 20, 26, 27 (in this order). Start by using the lowest 2 bits for the hash function. Show each step. (10 pts)

2. Consider the following Linear Hash Table, and assume that a bucket split occurs whenever an overflow page is created. h0(x) takes the rightmost 2 bits of key x as the hash value, and h1(x) takes the rightmost 3 bits of key x as the hash value.
   Now insert 13, 19 and 20 and draw the final hash table. Write the new hash function (if any), and also update the “Next” pointer. (10 pts)



3. Let us consider a relation R (s, t, u) containing 1 billion (10^9) records. Each page of the relation holds 10 records. R is organized as a heap file with unclustered indexes and the records in R are randomly ordered. Assume that atribute ***s*** is a candidate key for R, with value lying in range 0 to 999,999,999.
   Now, for each of the following queries, name the approach that would most likely require the fewest I/Os for processing the query. Justify your answer by approximately calculating the cost (in terms of disk accesses) for each approach to answer each query. (20 pts)

   The approaches to consider are as follows:

   i. Scanning through the whole heap file for R.
   ii. Using a B+ tree index on attribute R.s
   iii. Using a hash index on attribute R.s

   The queries are:

   i. Find all R tuples.
   ii. Find all R tuples where R.s < 100
   iii. Find all R tuples where R.s = 100

   iv. Find all R tuples where R.s >100 and R.s<150

Also assume that the height of the B+Tree is: **h=3**, the number of data entries per page is **M and M>10**. Additionally, (assume that) after accessing the data entry, it takes one more disk access to get the actual record and let the occupancy factor in hash index is: ***c***.