CS3223: Database Management Systems

Tutorial 2

(Week 4, Feb 2022)

1. Consider a relation EMP(eno, sno, name, salary) containing 1,000,000 employee records. Suppose each page of the relation contains 10 records, and relation EMP is organized as a sorted file. Suppose sno is a candidate key, with values lying in the range 0-999,999, and that EMP is stored in sno order. Assume that each page is 4K bytes, eno and sno are each 4 bytes, and a pointer to a page is also 4 bytes. For each of the following queries, state which of the following three approaches is most likely to be the cheapest (in terms of I/O count). Justify your answer. For simplicity, assume that the data are uniformly distributed in the domain.

   - Scan the sorted file for EMP.
   - Use a (clustered) B+-tree index on sno. Assume that the nodes are 100% full, and that format 2 is used for the index.
   - Use a hash index on sno. You may assume ideal situation of no overflows, and each bucket is 70% full (and a minimum number of buckets is used). You may assume that format 1 is used for the index.

   a) Find all employees whose sno < 100,000.
   b) Find the employee whose sno = 10.
   c) Find the employees whose sno is in the range 10,000-10,010.
   d) Find the employees whose sno is not 10.

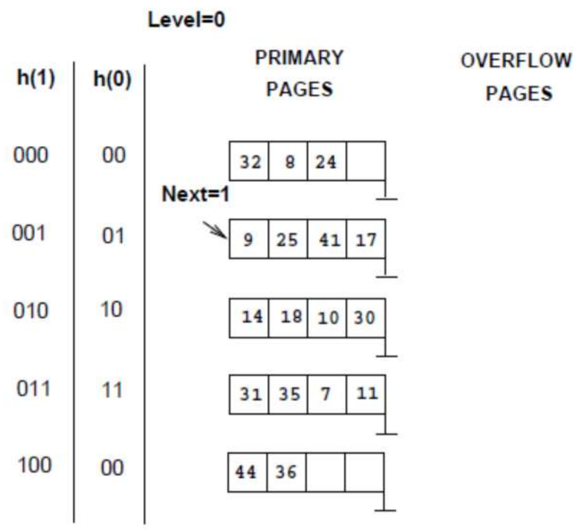
2. Consider a relation Book(Bookid: integer, Author: string, …) where Bookid is the key, and its values are assigned in increasing order starting from 1. Suppose Book has 6 million records of 200 bytes each. Suppose we have to perform 10,000 single-record accesses, and 100 range queries of 0.005% of the file, with Bookid as the search key.

   - Use hashing (with key-to-address transformation of the form x mod y). Suppose the hash table has a load factor of 70% and the bucket size is 4096 bytes. Moreover, assume that records are stored in the bucket (format 1), and there is no overflow of buckets.
   - Use B+-tree. Suppose each node is 70% full, and the sizes of a node, key and address are 4096, 8 and 4 bytes respectively.

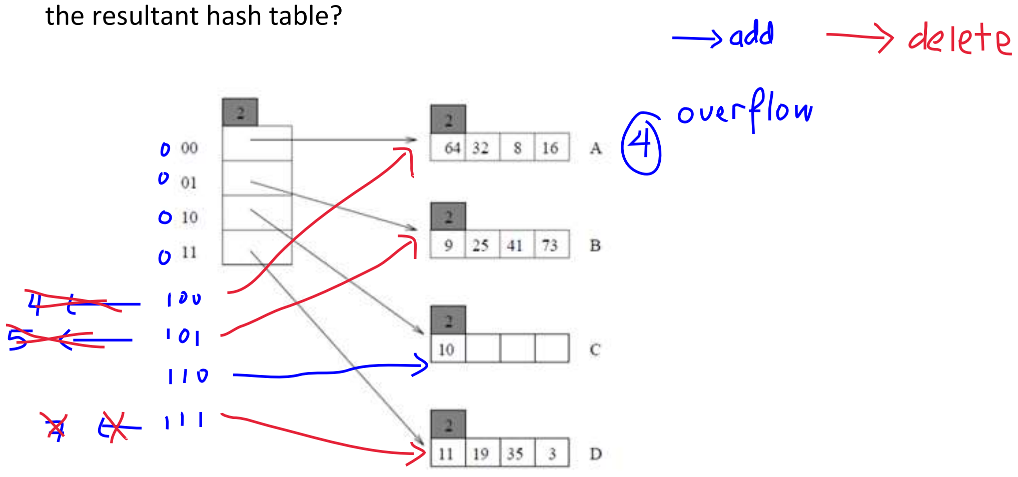
   Which of the above two methods is better for the application? Under what circumstance will the "loser" outperform the "winner"?

3. (Question taken from the main reference text – Ex 11.2) Consider the Linear Hashing index shown in the figure below. Assume that we split whenever an overflow page is created. Answer the following questions about this index:

a) What can you say about the last entry that was inserted into the index if you know that there have been no deletions from this index so far?
b) Show the index after inserting an entry with hash value 4, and 15.
c) Show the index after deleting the entries with hash values 36 and 44 into the original tree. Assume that the full deletion algorithm is used.
d) Find a list of entries whose insertion into the original index would lead to a bucket with two overflow pages. Use as few entries as possible to accomplish this.



4. (Adapted from question in the main reference text – Ex 11.11.4) Consider the extendible hash table below. Insert 4, 5 and 7. Show the resultant hash table? Now, delete 7, 5, and 4. What is the resultant hash table?



since now all the split image buckets point to the same bucket as their original directory, we can now half the directory