25 + 2

CS 564, Spring 2017: Quiz #1

You have 30 minutes to finish this quiz. The total number of points on this quiz is 30.

NAME: Kirthanaa Raghuraman *UW ID:* kraghuraman / 9073422751

Q1 [4 points]: Give an example of an access pattern where MRU is a better replacement policy than LRU

Consider a buffer pool of 3 frames and an access pattern of size 4 as follows: 1,2,3,4 & repeated scan from start to end
So, when we use LRU here, all the pages are thrashed and hence there will be a lot of I/Os. When it comes to MRU, the no of I/O is significantly reduced.

Q2 [6 points] Consider storing data on a hard disk drive with one disk platter, and one hundred tracks. Assume that the innermost track is labeled as track #0, and that the outermost track is labeled as track #99. If you have to allocate space for a file on an empty disk, and if this file is going be read sequentially often, which track would you use to start storing the file? Why?

I would use the outermost track (#99) for the following reasons:
1) It has the largest circumference among all tracks & hence can store lot of data.
2) The seek time (assuming the read/write head is at rest near the same track initially) will be the lowest for the outermost track.
3) I can store the file (if its large & doesn't fit in one track) in track #98, #97 .. and so on which have storage capacities in decreasing order.

Q3 [5 points]: In a typical slotted page organization, a record id is a pair **(page#, slot#)**. A friend of yours suggests changing the record id, replacing the slot# in the record id by the offset to the first byte of the record from the start of the page. Thus, a record id would be a pair **(page#, offset)**. Recall that this offset information is what is stored in the slot directory. Now assuming that the number of bytes needed to represent a slot# is the same as the offset, do you think this suggestion is a good idea? Explain your answer.

Your answer: □ Yes ☑ No

Explain your answer:

Assuming the record is NOT read only, using (page #, offset) as record id is a bad idea as if records grow/shrink, they may be rearranged and compacted => This leads to a change in the record ids as offset will change.

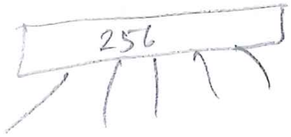
However, if the records are write-once and read only, this could be a good idea as it becomes equivalent to (page #, slot #) format.

No of entries in each node is atleast $256/2 = 128$. Root is exempt from this condition. In practice, non-leaf nodes are more packed than leaf nodes.
$\therefore \# L = 2^{20}/128 = 2^{13}$, #NL above leaf $= 2^{13}/128 = 64$, Total $= 2^{13}+64+1$

Q4: [6 points] Consider a relation with 2^{20} records (~1 million records). Assume that a page can hold at most 256 data entries (both in the leaf and the non-leaf nodes). Now consider a B+-tree index on this relation. In the *worst* case, how many pages would be used to store this index? Show how you compute your answer.

$F = 256+1$

Worst case: each index node can have only 'd' entries



Here $d = 128$

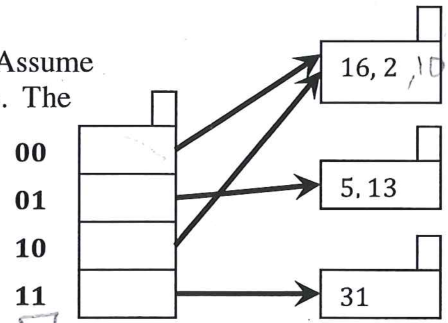
$\therefore$ No of index nodes $= \frac{2^{20}}{256/2} = 2^{11}$ → Should be 2^{13}

-3

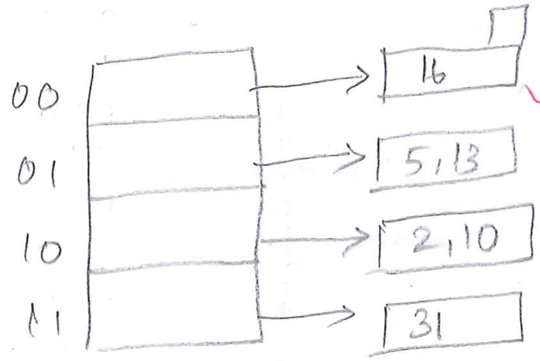
No of leaf nodes $= 2^{20}$ (# of records)

$\therefore$ No of pages (worst case) $= \frac{2^{31}}{2^{11}} = 2^{20}$ $\left(\frac{2^{\text{index+leaf nodes}}}{\text{\# records in page}}\right)$

Q5 [5 points] Consider the extendible hashed index file shown on the right. Assume that the hash function is h(key) = key mod 32, and that 2 records fit on a page. The figure below indicates the presence of records on a leaf page by listing the keys of the records on the page. Redraw the figure shown above to show the index structure after inserting a record with key 10.



Binary value of 10 = 1010
global depth = 2, ⇒ 10



16 = 10000
2 = 10
10 = 1010

Q6 [4 points] Answer True or False.

A Bitmap index has no record ids in the index file. ☑ True □ False

-2

The number of bytes in a Bitmap index file on a Boolean attribute (i.e. the attribute can take values Y and N) is the same as the number of tuples in the relation/table. (You can ignore null values for this question) ☑ True □ False

⇒ 2 x no of tuples

x2

On a Boolean attribute (again assume no NULLs) the Bitmap and Bitsliced index converge to become the same index structure. ☑ True □ False

Bitsliced indices are more compressible than Bitmap indices. □ True ☑ False

files needed to store this = 2 (one for T column, one for F column)