TU München, Fakultät für Informatik
Lehrstuhl III: Datenbanksysteme
Prof. Alfons Kemper, Ph.D.

Exercise for *Database System Concepts for Non-Computer Scientist* im WiSe 19/20

Alexander van Renen (renen@in.tum.de)
http://db.in.tum.de/teaching/ws1920/DBSandere/?lang=en

Sheet 13

Exercise 1

[Same as in the previous sheet, but with a B+-Tree, instead of a B-Tree.] Calculate the optimal degree i and the number of required levels (also known as the "height" of the tree) for a B+-Tree with the following properties:

- The B+-Tree should store all humans currently living on earth (assume an even 10 billion).
- For each human we store the name, country and a unique identifier (100 Byte per human). The unique identifier will be used as the key an requires 8 Byte to store.
- The degree i of inner and leaf nodes may be different.
- Each node has to fit on a 16KB (16000 Byte) page.
- The page ids in the inner nodes require 8 Byte.
- This time (unlike in the lecture), we want to be precise: an inner node with n tuples requires $n+1$ page ids to identify its children (in the lecture we simplifies this and assumed that a node with n tuples has n page ids).

Solution:

For leaf nodes, we simply have to store the tuples themselves and we can calculate the number of tuples fitting on a single leaf node as follows: leaf_size $\div$ tuple_size $= 16\text{KB} \div 100\text{B} = 160$. The degree of a node is half of that: 80. Using this, we can calculate the number of leaf nodes required to store 10 billion human tuples: number_of_humans $\div$ tuples_per_leaf $= 10e9 \div 160 = 62500000$. So far, nothing has changed compared to the B-Tree.

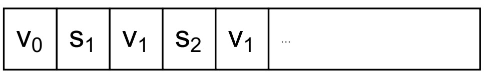


Figure 1: B+-Tree node structure

Next we calculate how many separator keys (x) can fit on an inner node. From this we can derive the fan-out of an inner node (how many pages can be addresses by an inner node). Using the structure of an inner node (Figure 1), we can create the following formula:

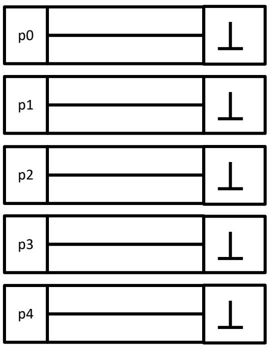
$$
\begin{aligned}
x * \text{key\_size} + (x+1) * \text{page\_id\_size} &\leq 16\text{KB} \\
x * 8\text{B} + (x+1) * 8\text{B} &\leq 16\text{KB} \\
x * 8\text{B} + x * 8\text{B} + 8\text{B} &\leq 16\text{KB} \\
x * 16\text{B} &\leq 16\text{KB} - 8\text{B} \\
x &\leq (16\text{KB} - 8\text{B}) \div 16\text{B} = 999.5
\end{aligned}
$$

Hence, we can store 999 separator keys in an inner node (because we need to round up, because we only store "complete" tuples) and can therefore address $999 + 1 = 1000$ child pages. Therefore, a total of $62500000 \div 1000 = 62500$ inner pages are required to store each page on the leaf level. But these 62500 pages need to be addresses as well ...

To figure out the height of the tree (number of layers, not counting the root), we can either continuously divide until there is only one page left: $62500000 \div 1000 \div 1000 \div 1000 = 0.0625$ and see that there is one leaf level and three layers of inner nodes. Or, we can use a logarithm: $\log_{1000}(62500000) \approx 2.3$ to derive the number of inner layers (rounded up: 3). In both cases we end up with 4 layers (3 inner, 1 leaf). Therefore, the tree has a "height" of 3, because the root does not count.

Exercise 2

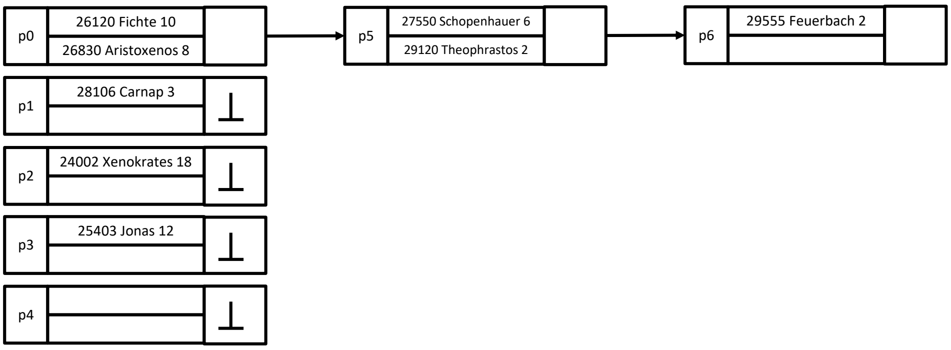
Please insert all tuples from the `Students` relation from the university schema into a hash table of size 5 (as in the figure). Each page can hold up to 2 tuples. As a means of handling collisions, linear chaining should be employed.



a) Use the following hash function: hash(key) = key mod 5.
b) Try using a better hash function: hash(key) = crc32(key) mod 5 To calculate the CRC32 of a key, you can use a website on the internet, for example: `https://crccalc.com/?crc=24002&method=crc32&datatype=ascii&outtype=dec`

   Did the better hash function, result in a more evenly balanced hash table?

Solution:


p0
26120 Fichte 10
26830 Aristoxenos 8
p5
27550 Schopenhauer 6
29120 Theophrastos 2
p6
29555 Feuerbach 2
p1
28106 Carnap 3
p2
24002 Xenokrates 18
p3
25403 Jonas 12
p4