Exercise for *Database System Concepts for Non-Computer Scientist* im WiSe 18/19

Alexander van Renen (renen@in.tum.de)
http://db.in.tum.de/teaching/ws1819/DBSandere/?lang=en

Sheet 11

Exercise 1

Consider the *Students* relation in our university schema:

a) Estimate the size (in byte) for an average row of the *Students* table and then calculate the size of the entire table, assuming there are 1 million students.

b) Calculate the time it would take read the entire table using sequential and random access using an HDD. The seek time is 4 milli seconds, the time for the rotation is 2 milli seconds, and the transfer time is 0.1 milli seconds for a 4 KB block of data (You can ignore the track-to-track-seek-time). You always have to transfer an entire 4KB page, even if you are only reading a single student tuple (in the random access case).

c) Try out the same calculation for an SSD. Here we do not have any seek or rotation time. In addition, the transfer time is as low as 10 micro seconds for reading a 4 KB page.

[Bonus] This exercise is very optional and only if you want to test out your programming skills. Try to write a small C or C++ program to measure the performance difference between random and sequential read of the storage devices on your machine. For this simply create a big file (around 1GB) by writing a bunch of data to your SSD or HDD (you can use malloc to obtain 1GB of memory and then initialize it to any random number). Next, read this data back in: once sequentially from begin to end and once in a random order. Hint: You can use fopen and fclose to open and close a file. And then the fread and fwrite functions to read and write data. For moving around in a file there is the fseek function.

Solution:

a) A row in the *Students* consists of two *integer* columns (4 Byte) and one *varchar* column with length 30 (30 Byte). We, therefore, estimate the size of a student tuple with 38 Byte. Using this, we can calculate the size of the table: 38 Byte$*$1000000 = 38 MB

Note: In practice, a database system would usually need a little bit more space to store the tuple: The length of the text in the *varchar* field needs to be stored. There needs to be a way to determine whether the *semester* is *null*. And a few other implementation artifacts increase the tuple size. But for the purpose of the exercise we can assume 38 Byte.

b) **Sequential access:** We have to move the read-write-head of the HDD to the correct position and wait for the disc to rotate to the right position. After that we can simply read the 38 MB without repositioning the read-write-head (it only needs to move to neighboring tracks, which we ignore in this exercise). Due to the block-based nature of the device (it can only transfer 4 KB blocks of data), we might have to read some additional bytes at the end:

$\text{seek\_time} + \text{rotation\_time} + \text{transfer\_time} * \lceil 38 \text{ MB} \div 4 \text{ KB} \rceil$

$= 4 \text{ ms} + 2 \text{ ms} + 0.1 \text{ ms} * \lceil 38 \text{ MB} \div 4 \text{ KB} \rceil$

$= 956 \text{ ms}$

Random access: This time we need to reposition the read-write-head and wait for the disc to spin to the right position for each tuple we transfer. In addition, (as above) we can not only transfer 38 Byte, but need to read an entire 4 KB block (or page) from the HDD. Thus, instead of reading 38 Byte a million times, we actually read 4 KB a million times (we ignore the unlikely cases where two subsequent accesses to a tuple happen to be located on the same 4 KB block/page):

$(\text{seek\_time} + \text{rotation\_time} + \text{transfer\_time}) * 1000000$

$= (4 \text{ ms} + 2 \text{ ms} + 0.1 \text{ ms}) * 1000000$

$= 6100 \text{ s} \approx 1 \text{ h}40 \text{ m}$

c) For SSDs we can do the same calculation, but simply set the seek and rotation time to zero and adjust the transfer time:

Sequential access:

$\text{transfer\_time} * \lceil 38 \text{ MB} \div 4 \text{ KB} \rceil$

$= 10 \ \mu\text{s} * \lceil 38 \text{ MB} \div 4 \text{ KB} \rceil$

$= 95 \text{ ms}$

Random access:

$(\text{transfer\_time}) * 1000000$

$= (10 \ \mu\text{s}) * 1000000$

$= 10 \text{ s}$