Prof. Dr. Stefan Brass
Institut für Informatik
MLU Halle-Wittenberg

November 28, 2019

Databases IIB: DBMS-Implementation
— Exercise Sheet 6 —

Please read Part a) and mark questions which you want to discuss in class. You only have to submit Part b) and c). Please upload your solution (a pure ASCII text file with the two SQL statements) into the StudIP file folder called "Hausaufgabe_6" in the StudIP entry of the lecture. The deadline is December 3 (the day before the next lecture).

It is permitted to form groups of up two members, but please make sure that both members can fully explain all homeworks submitted by the group. Please upload only one file per group. Name the file such that it contains the names of all group members.

Not all submitted homeworks will be corrected, but all homework exercises will be discussed in class. If you should have questions about your homework, please ask! A precondition for getting credit for this course is that you submit solutions to two thirds of the homeworks. Obviously wrong or very incomplete submissions do not count.

Repetition Questions

a) What would you answer to the following questions in an oral exam?

- Name some background processes of Oracle and explain their purpose. In order to be not too Oracle-specific, think about tasks a DBMS should do asynchronously, i.e. not as part of the process or thread that executes a query or update.
- Name some data structures in the shared memory of all Oracle processes (the System Global Area). Again, try to think more generally about any DBMS that executes several queries in different processes/threads at the same time. What data should they share? Even if you have a server that executes only one query at a time, what data should it keep in memory from one query to the next?
- What is the purpose of a checkpoint?
- What is the difference between the dedicated server architecture and the multi-threaded server architecture of Oracle?
- What are the main components of a disk (hard disk drive)?
- What is a sector? How can one specify the position of a single sector on the disk, i.e. what are the components of a sector address?
- What is a typical sector size? Why do many DBMS and operating systems use a larger block size ("cluster size" on Windows)? What is a typical block size? Discuss advantages/disadvantages of increasing the block size.

- Why do some disks have more sectors on the outer tracks than on the inner tracks?
- If a power failure happens, is it possible that a block is only partially written?
- Why is using RAM in the disk controller for "read ahead" a good idea?
- How long does a typical hard disk drive need to read or write one block of data? (Obviously, there is not one exact value that is correct. However, you should know the order of magnitude.)
- How many random block accesses are possible per second with a typical hard disk drive?
- How many bytes can be read per second sequentially from a typical hard disk drive?
- What are typical rotational speeds of hard disks? Which part of the access time for a block can be improved with higher rotational speeds?
- Compare an SSD with a standard magnetic hard disk drive.

Homework Exercise 6

b) Please either find out which disk is installed on your own PC, or name some disk you would like in your next PC. Furthermore, try to find some data on that disk in the internet. Any three data items (numbers) would suffice (at least the disk capacity you should know, maybe you can find e.g. the seek time and the rotation speed in addition). If you cannot find data on your disk, choose any other disk.

On Windows, the name of the disk should be listed in the "Device Manager", which you can reach via "Settings", "System", "about", and then it is linked at the bottom. For Linux, you find some information on the following pages:

- [https://www.cyberciti.biz/faq/find-hard-disk-hardware-specs-on-linux/]
- [https://unix.stackexchange.com/questions/4561/]

Some places for getting performance comparisons of disks are:

- [http://www.storagereview.com/]
- [http://www.tomshardware.com/]

Of course, you can also check the web pages of big disk manufacturers for data sheets of their disks, e.g.

- [https://www.seagate.com/internal-hard-drives/hdd/] (English)
  [https://www.seagate.com/de/de/internal-hard-drives/hdd/] (German)
- [https://www.hgst.com/products/hard-drives] (English)
  [https://www.hgst.com/de/products/hard-drives] (German)

- [https://www.wdc.com/products/internal-storage.html] (English)
[https://www.wdc.com/de-de/products/internal-storage.html] (German)
- [https://en.wikipedia.org/wiki/List_of_computer_hardware_manufacturers]

If you wish, you can also use a disk benchmark (this is not required to finish this exercise, we do our own small measurement in the next exercise):

- [https://www.heise.de/download/product/crystaldiskmark-46961]
- [https://crystalmark.info/en/software/crystaldiskmark/]
- [https://www.chip.de/downloads/ATTO-Disk-Benchmark_41802004.html]
- [https://www.atto.com/disk-benchmark/]
- [https://www.thomas-krenn.com/de/wiki/Fio]

c) Write a program that reads or writes a relatively large file (at least 10 MB). Measure the runtime of this program and compute from that how many MB could be read or written per second. Note that when reading a file, you must measure the time when you access the file first. After that, the disk blocks will be in the cache managed by the operating system, and reading will be much quicker. When writing, you must make sure that the data is really written to disk, and not yet in the OS cache.

[https://stackoverflow.com/questions/13358431/force-write-of-a-file-to-disk]

Measuring Runtime in C++

- You can measure the runtime in your program, and also from outside. Both is interesting: If you do the measuring in your program, you can distinguish different steps of your program. You can also process the results in your program, e.g. easily format them as you like and integrate them into other output of your program.

  If you do the measurements from outside, you get the entire runtime, including startup. Furthermore, it protects you from suspicions that maybe you did the measurements in your own program incorrectly.

- In Linux, `/usr/bin/time` is a tool for measuring runtime from outside. One calls it in the form

  ```
  /usr/bin/time ./myprog
  ```

  if you would normally run your program as `./myprog`. It shows the CPU time in user mode and in system mode (for operating system calls), and also the elapsed time (real time/wallclock time). There is also information about how much RAM was used and more information about inputs, outputs and page faults. The exact output is configurable. Note that when you simply use `time` in `bash` (instead of `/usr/bin/time`), you get a simpler version built into the shell. However, for our purposes, this is sufficient. You can get information about the currently running programs with `top` and `ps`. If you want to get a lot of information, e.g. about memory usage of a process, you can access the process file system:

  [https://en.wikipedia.org/wiki/Procfs]

- Under Windows, one can use `Measure-Command` in the PowerShell, e.g.

  ```
  Measure-Command { copy x y }
  ```

  See:

  [https://docs.microsoft.com/en-us/powershell/module/]
  [microsoft.powershell.utility/measure-command?view=powershell-6]

- For measuring within your program, have a look at e.g.

  [https://www.geeksforgeeks.org/]
  [measure-execution-time-with-high-precision-in-c-c/]

  or

  [http://www.willemer.de/informatik/cpp/timelib.htm]

  From C++ 2011, the `chrono` library was introduced:

  [https://en.cppreference.com/w/cpp/chrono]

- With older C++ versions, solutions are system-dependent. Under Linux, one can use `clock()` for CPU time:

  [http://www.cplusplus.com/reference/ctime/clock/]

  For real time, one can use `gettimeofday`:

  [http://rabbit.eng.miami.edu/info/functions/time.html]

  One must include `<sys/time.h>` and `<time.h>`.

- For Windows, CPU time can be measured with `GetProcessTimes`:

  [https://docs.microsoft.com/en-us/windows/win32/api/]
  [processthreadsapi/nf-processthreadsapi-getprocesstimes]

  Real time can be measured with `GetSystemTimeAsFileTime`:

  [https://docs.microsoft.com/en-us/windows/win32/api/]
  [sysinfoapi/nf-sysinfoapi-getsystemtimeasfiletime]

  One must include `<windows.h>` and `<time.h>`.