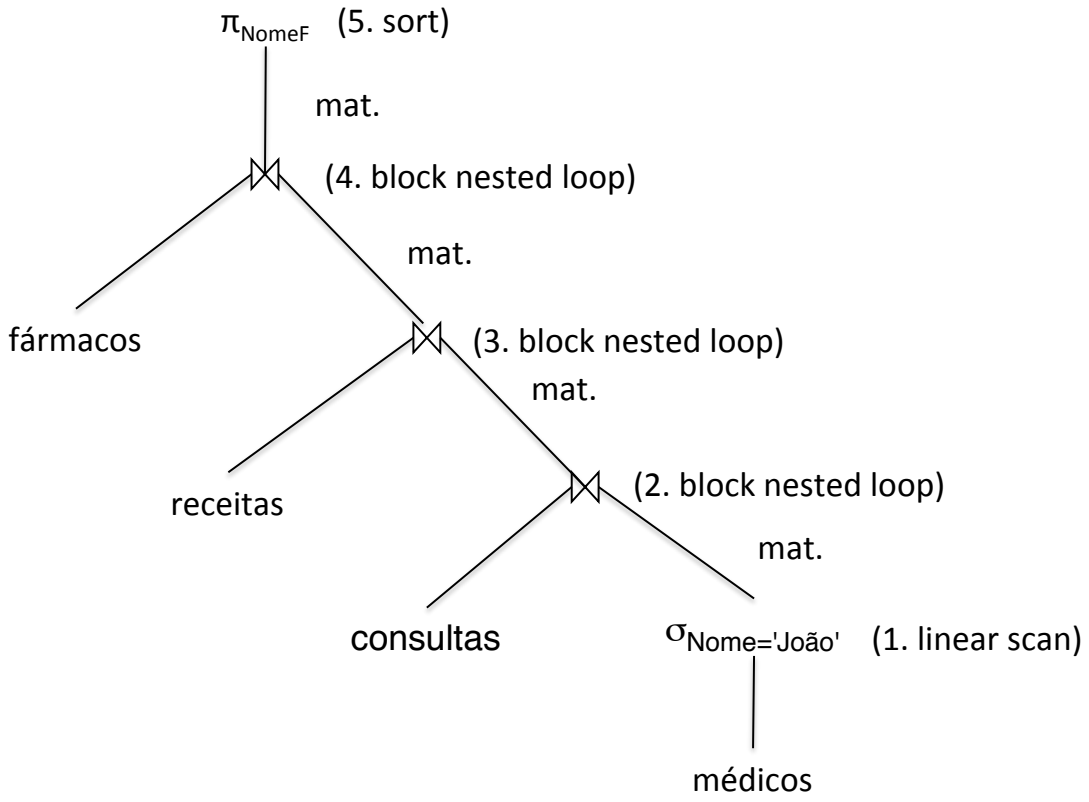
Detailed analysis for answering to questions involving estimates of size of joins and space requirements (see exercise 1b of 2012/13 midterm).

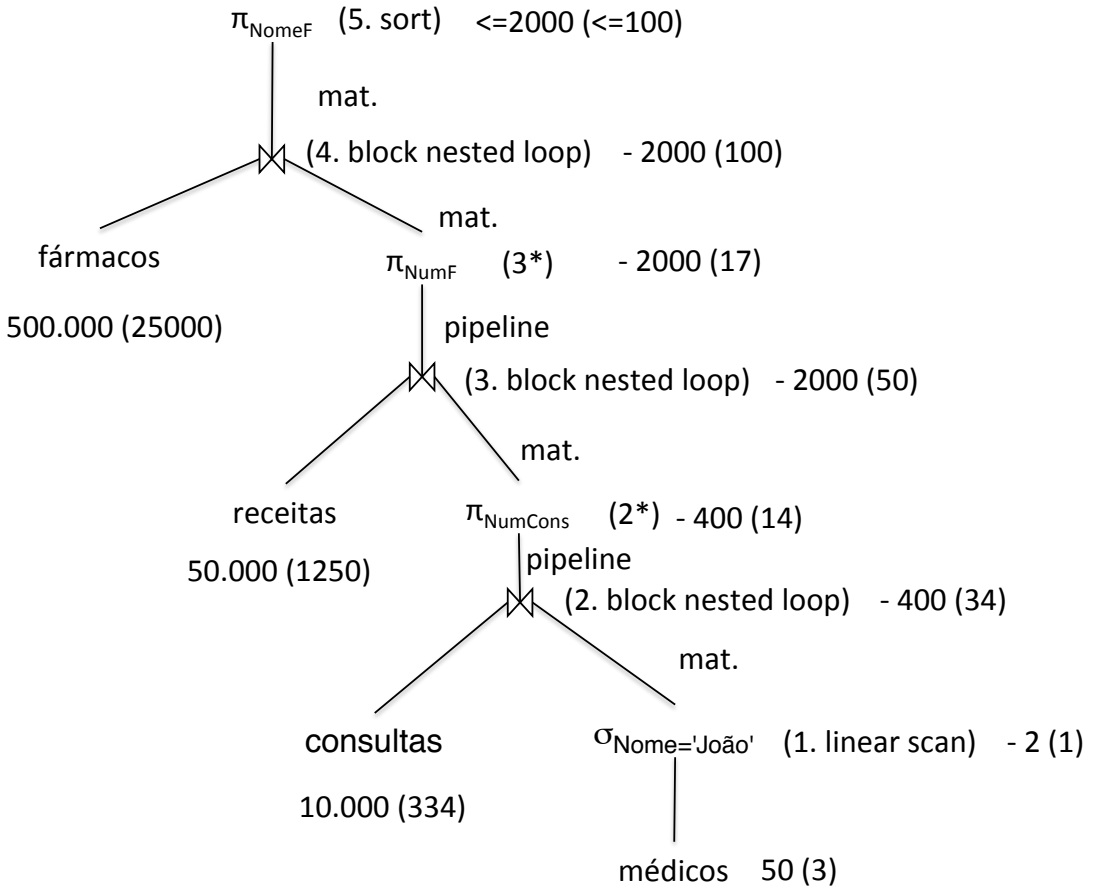


1.) Assuming that we have two doctors called 'João' in table medicos, the join will return 2 tuples each occupying 1 / 20 of a block. So, we will need at most one block to store the result of the selection. The complete selection operation will require 3 block transfers and 1 seek to scan the table, and will require 2 block transfers (because writing a block costs the double of a block transfer) and 1 seek to materialize the result.

2.) To perform the join we will require 334 + 1 block transfers and 2 seeks, since the smallest relation fits in memory. It will produce 400 tuples, since in average every doctor performs 200 appointments (10000/50) and every appointment must be performed exactly by one doctor.

Regarding the space necessary to store the result note that each in total it will be required the information of 400 tuples of consultas + 400 tuples of medicos (with repeated information, of course). So, we will need respectively 13.333 (400/30) + 20 (400/20) = 33.333 blocks, i.e. 34 blocks. However, notice that in fact we will need only to know NumCons to execute the rest of the query. Therefore, in the worst case we will require only 14 blocks (assuming that NumCons occupies all the space of a consultas tuple, which is of course not the case but it is an upper bound). This is clear in the following equivalent plan where we project the result of some joins without removing duplicates. Therefore, we require more 28 block transfers plus 1 seek to materialize the

result. Each operation is additionally annotated with the number of tuples (blocks) returned.



3) The execution of the join costs 1250 + 14 block transfers and 2 seeks. It produces 2000 tuples occupying 50 blocks. Note that NumCons is the only attribute of the inner relation which is also the join attribute, thus this join produces exactly tuples of the receitas schema. Therefore, 2000 tuples occupy 50 blocks because the blocking factor of receitas is 40. The projection in (3*) will reduce the blocks to one third since the receitas table has 3 integer attributes.
The materialization will cost 34 block transfers plus 1 seek.

Note that if we did not introduce the projections (2*) and (3*) in the plan this join would produce a relation requiring space to store 2000 receitas + 2000 tuples of join 2, taking 50 + 34*5 (why?) = 220 blocks that would no longer fit in memory when performing join 4.

4) This join requires 25.000 + 17 block transfers and 2 seeks. It produces 2000 tuples of the fármacos relation since the join is with a relation having only the join attribute (see above discussion). The materialization costs 200 block transfers and 1 seek.

5) Luckily, the result of join 4 fits in memory, and thus we only need 100 block transfers and 1 seek to read all the relation in memory and sort it in memory. We

don't know how many duplicates we have, so in the worst case we produce exactly the names of all medicines.

An estimate of the cost of the whole plan can be found in the next table.

Operation	**Block Transfers (t_T)**	**Seeks (t_S)**
1	3	1
Mat. of 1	2*1=2	1
2 and 2*	334+1=335	2
Mat. of 2*	2*14=28	1
3 and 3*	1250+14=1264	2
Mat. of 3*	2*17=34	1
4	25000+17=25017	2
Mat. of 4	2*100=200	1
5	100	1
Total	$26983t_T$	$12t_S$

The use of an indexed block nested-loop join does not pay off for performing operation 4. Assuming that $t_S=10t_T$ and that the height of the B⁺-tree of the index of the primary key of fármacos table is 3, the estimated cost is

$17*(t_T + t_S) + 2000*(3+1)*(t_T + t_S)= 17*(t_T + 10t_T) + 8000*(t_T + 10t_T)= 88171t_T$

constrasting with $26983t_T+12t_S = 26983t_T + 120t_T = 27103t_T$ of the initial plan.