

Selected exercises

Advanced database technology

Spring 2004

The below exercises were posed in the 2003 edition of the course, and are also relevant for the present edition. If you would like to discuss some of these exercises, or exercises posed during the course, at the June 3 exercise class, please send questions to anna@itu.dk no later than May 31.

The problems “most similar” to what may appear at an exam are marked by (E). Most problems are from the first half of the course. Note that some problems in G UW have solutions available on the web.

1. G UW 5.2.8 on page 213 (for the operators mentioned in the first lecture). Consider how monotonicity could be used when maintaining the results of relational expressions during updates of the underlying relations (known as “materialized views”).
2. (E) G UW 5.3.4 b+g+h on page 220–221.
3. (E) G UW 5.3.5 b on page 221.
4. G UW 11.3.2
5. (E) G UW 11.4.2
6. (E) G UW 11.4.8
7. G UW 12.3.11 on page 589.
8. (E) G UW 12.5.1 on page 601-2.
9. G UW 12.5.2 on page 602.
10. We have discussed *global rebuilding* of a database, which eliminates all space waste due to tombstones. The problem of updating the pointers was left open. Assume that an entire database using D blocks of disk space has been rebuilt, except for its pointers. With every record, the previous position of that record is stored. Show how to efficiently update all pointers to old record locations to point to the new locations. What is the I/O complexity of updating the pointers? **Hint:** Use sorting (several times) to pair pointers and positions.
11. G UW 13.3.8.
12. Suppose that two keys K_1 and K_2 are Z keys apart in the sorted order of N keys. Show that a search for K_1 **and** K_2 can be done in $O(\log_n N + \log_n Z)$ I/Os, by using the B⁺-tree of G UW Exercise 13.3.8.
13. G UW 13.4.3.
14. In a problem session you have found bad key sets (resulting in long chains) for some specific hash functions. Argue that any fixed hash function will have bad key sets if the number of possible keys is large enough. Why does it help to pick a function at random?

15. Consider linear hashing where we want to keep the space utilization $\alpha = 1/2$. Suppose that B is so large that we have no block overflows. What is the number of I/Os needed for inserting N keys in a table of initial size 1?
16. Now consider the above question for uniform rehashing, assuming that whenever the space utilization is above $\alpha = 2/3$ we rehash to a table of twice the size.
17. (E) G UW 15.3.4.
18. (E) G UW 15.5.4.
19. (E) G UW 15.7.2.
20. G UW 16.3.3 on page 820.
21. G UW 16.4.3 on page 835.
22. (E) G UW 16.7.4 on page 872.