Implementation of DBMS
Exercise Sheet 5, Solutions
Klingemann, WS 2025 / 2026

1) Suppose that we have 4096-byte blocks in which we store records of 100 bytes. The block header consists of an offset table using 2-byte pointers to records within the block. Each day one record is deleted (if records are in the block) and afterwards two records are inserted. A deleted record must have its pointer in the offset table replaced by a tombstone. If the block is initially empty, for how many days can we insert records into a block?

Solution:

For the new records we insert each day we need (including the corresponding entries in the offset table) 100 + 2 + 100 + 2 bytes = 204 bytes. On day one, we use exactly this number of bytes, as we have no records to delete. On subsequent days, we save 100 bytes from the deleted record. However, we cannot reuse the 2 bytes from the offset table. As a result, starting from day 2, we need 204 – 100 bytes = 104 bytes each day. Therefore, the number of days we can insert the records using the given pattern is:

1 day + $\lfloor$(4096 – 204) bytes / (104 bytes / day)$\rfloor$ = 38 days

2) Suppose that if we swizzle all pointers automatically, we can perform the swizzling in half the time it would take to swizzle each one separately. If the probability that a pointer in main memory will be followed at least once is `p`, for what values of `p` is it more efficient to swizzle automatically than on demand?

Solution:

Let `t` be the time to swizzle a pointer on demand. When we swizzle on demand, we have to invest this amount of time with a probability of `p`. Therefore, the average amount of time per pointer is `pt`. When we swizzle automatically, we have to invest for each pointer `t/2`. As a result, swizzling automatically is more efficient when `t/2 < pt`, i.e., `p > ½`.

3) Suppose blocks hold either three records or ten key-pointer pairs. As a function of n, the number of records, at least how many blocks do we need to hold

a) the data file

b) a dense index

c) a sparse index

You can ignore inaccuracies that result from rounding.

Solutions:

a) (n records) / (3 records / block) = n/3 blocks

b) (n key-pointer pairs) / (10 key-pointer pairs / block) = n/10 blocks

c) (n/3 key-pointer pairs) / (10 key-pointer pairs / block) = n/30 blocks

4) Suppose blocks hold either ten records, or 50 key-pointer pairs. We have a data file with 10^6 records. In this task we assume that each block will be as full as possible.

a) How many blocks do we need for the data file?

b) How many blocks do we need for a dense index?

c) How many blocks do we need for a sparse index?

d) We want to add higher level index structures to the index considered in b). Describe how many blocks we have in these higher levels until we end up in a level with just one block.

e) Repeat d) for the sparse index in c).

Solutions:

a) $\lceil (10^6 \text{ records}) / (10 \text{ records / block}) \rceil = 10^5$ blocks

b) $\lceil (10^6 \text{ key-pointer pairs}) / (50 \text{ key-pointer pairs / block}) \rceil = 20000$ blocks

c) $\lceil (10^5 \text{ key-pointer pairs}) / (50 \text{ key-pointer pairs / block}) \rceil = 2000$ blocks

d) 2nd level index: $\lceil (20000 \text{ key-pointer pairs}) / (50 \text{ key-pointer pairs / block}) \rceil = 400$ blocks
   3rd level index: $\lceil (400 \text{ key-pointer pairs}) / (50 \text{ key-pointer pairs / block}) \rceil = 8$ blocks
   4th level index: $\lceil (8 \text{ key-pointer pairs}) / (50 \text{ key-pointer pairs / block}) \rceil = 1$ block

e) 2nd level index: $\lceil (2000 \text{ key-pointer pairs}) / (50 \text{ key-pointer pairs / block}) \rceil = 40$ blocks
   3rd level index: $\lceil (40 \text{ key-pointer pairs}) / (50 \text{ key-pointer pairs / block}) \rceil = 1$ block