Implementation of DBMS
Exercise Sheet 4, Solutions
Klingemann, WS 2024 / 2025

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) We have a data file with 10^4 records. Records and blocks are like in task 3a) of Sheet 3. How many blocks do we need for the data file? records = 48 bytes / record and a block of 4096 bytes

a) We use spanned storage.

b) We use unspanned storage.

Solution:

a) In total we need 48 bytes / record * 10000 records = 480,000 bytes. As we have a block header that consumes 40 bytes, the number of bytes in a block available for storing records is 4096 bytes – 40 bytes = 4056 bytes. Due to spanned storage, we can use each of these bytes and need therefore

$\lceil$480,000 bytes / (4056 bytes for records / block)$\rceil$ = 119 blocks

b) We have already calculated in Sheet 3 that we can store 84 records in each block. Therefore, we need $\lceil$10000 records / (84 records / block)$\rceil$ = 120 blocks

3) Suppose that we handle insertions into a sequential data file of n records by creating overflow blocks as needed. Also, suppose that the data blocks are currently all half full. If we insert new records at random, how many records do we have to insert before the average number of data blocks (including overflow blocks if necessary) that we need to examine to find a record with a given key reaches 2? Assume that on a lookup, we search the block pointed to by the index first, and only search overflow blocks, in order, until we find the record, which is definitely in one of the blocks of the chain.

Solution:

To calculate the average number of data blocks that we need to examine, we have to take into account which portion of all records in a chain of blocks resides in a particular block. Initially we have just the blocks that we used when creating the file (the primary blocks). When we need more space, we start creating overflow blocks. For example, assume that we have already inserted some records so that for every primary block we have one overflow bock which is half full. Then the portion of records that is in the primary block is 2/3 and for the overflow block 1/3. For finding a record in the primary bock, we need to examine just one block. For finding a record in the overflow block we need to examine two blocks. Therefore, the average value is:

2/3 * 1 examined block + 1/3 * 2 examined blocks = 4/3 examined blocks.

When we continue inserting, we can see that we get an average of 2 when we have for every primary block two completely full overflow blocks as we get in this case:

1/3 * 1 examined block + 1/3 * 2 examined blocks + 1/3 * 3 examined blocks =
2 examined blocks.

As n records mean half full primary blocks, we have in this scenario 6n records. Thus, we added 5n records.

n records in the primary blocks (now completely full).
2n records in the two overflow blocks (both completely full).
Total records: = $n + 2n = 3n$
.
Thus, 5n records are inserted to reach an average of 2 blocks examined.

4) 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