Implementation of DBMS
Exercise Sheet 3, Solutions
Klingemann, WS 2025 / 2026

1) Suppose a record has no header but the following fields in this order: A character string of length 25 bytes, an integer of 2 bytes, an SQL date (requires 10 bytes), and an SQL time (no decimal point, requires 8 bytes). How many bytes does the record take if
a) Fields can start at any byte.
Solution:
We can simply add up the number of bytes required for the different fields:
25 + 2 + 10 + 8 bytes = 45 bytes
b) Fields must start at a byte that is a multiple of 4.
Solution:
If the number of bytes required for a field is not yet a multiple of 4, we have to assign the next larger multiple of 4 and get: 28 + 4 + 12 + 8 bytes = 52 bytes
c) Fields must start at a byte that is a multiple of 8.
Solution:
Similar as for b) but using a multiple of 8 we get: 32 + 8 + 16 + 8 bytes = 64 bytes

2) Assume the fields are as in Exercise 1, but the records also include a header consisting of two 4-byte integers and a 1-byte character. Calculate the record length for the three situations regarding field alignment a) through c).
Solution:
We do the same as in task 1) but include the fields of the header:
a) 4 + 4 + 1 + 45 bytes = 54 bytes
b) 4 + 4 + 4 + 52 bytes = 64 bytes
c) 8 + 8 + 8 + 64 bytes = 88 bytes

3) Suppose records are as in Exercise 2, and we wish to pack as many records as we can into a block of 4096 bytes, using a block header that consists of ten 4-byte integers. How many records can we fit in the block in each of the three situations regarding field alignment a) through c)? We use unspanned storage of records, i.e., a record must not be divided but stored completely within one block.
Solution:
Part of the block is needed for the block header and the rest can be used to store records. For a) and b) we need 10 * 4 bytes = 40 bytes for the header. For c) we need 10 * 8 bytes = 80 bytes. This gives us:
a) $\lfloor$(4096 – 40) bytes / (54 bytes / record)$\rfloor$ = 75 records
b) $\lfloor$(4096 – 40) bytes / (64 bytes / record)$\rfloor$ = 63 records
c) $\lfloor$(4096 – 80) bytes / (88 bytes / record)$\rfloor$ = 45 records

4) A patient record consists of the following fixed-length fields: the patient's date of birth, social-security number, and patient ID, each 10 bytes long. It also has the following variable-length fields: name, address, and patient history. If each pointer within a record requires 4 bytes, and a field for the record-length is a 4-byte integer, how many bytes, exclusive of the space needed for the variable-length fields, are needed for the record? The record header should consist of the record length and pointers to the variable-length fields. You may assume that no alignment of fields is required. Can you further reduce the record size by optimising its internal organization?
Solution:
The specification above tells us that for the record header we need a field for the record length and three pointers to variable length fields. Each consumes 4 bytes so that we get a record header of 16 bytes. When we add the three fixed-length fields of 10 bytes each, we get 46 bytes.
We can optimize this organization as one the pointers is not needed: The first variable-length field starts after the fixed-length part of the record so that we already know this position. With one pointer less, we need just 42 bytes.

5) Suppose records are as in Exercise 4) and the variable-length fields name, address, and history each have a length that is uniformly distributed. For the name the range is 10-50 bytes; for address it is 20-80 bytes, and for history it is 0-1000 bytes. What is the average length of a patient record?
Solution:
As the field-length' are uniformly distributed we know that the average length' of the corresponding fields are:
for name: (10 + 50) bytes / 2 = 30 bytes
for address: (20 + 80) bytes / 2 = 50 bytes
for history: (0 + 1000) bytes / 2 = 500 bytes
In total we have without the optimization of the header:
46 + 30 + 50 + 500 bytes = 626 bytes and with the optimization 4 bytes less, i.e., 622 bytes.

6) Suppose we have a relation whose n tuples each require R bytes, and we have a machine whose main memory M and disk-block-size are just sufficient to sort the n tuples using Two-Phase Multiway Merge Sort. How would the maximum n change if we made one of the following modifications of parameters?
a) Double B
b) Double R
c) Double M
Solution: We know that the maximum number of records is determined by M^2 / (R * B). As a result, for a) and b) we get half the number of records and for c) four times the number of records.