Database management, spring 2002 **Exercise 3 (15. – 19.4.)**

The minimum for active attendance: 3 tasks done A task marked with (**) are counted as two tasks.

1. (**) The EMPLOYEE relation has 10 000 records with the length of 100 characters. The schema of the relation is the same we have dealt with. The relation is implemented as a heap, and the page (block) size is 4 KB. The relation has two ISAM based secondary indexes: one for the attribute SSN (the attribute length 9 chars), and the other for the attribute ADDRESS (length = 40 chars). The bottom level of the indexes has the load factor of 80% (i.e., at the index creation time only 80% of the places for index records are used, 20 % is reserved for later insertions of new index records to eliminate the need for overflow records). (The basic index idea here is a generalization of Fig. 6.4 in E&N where only a single-level index is used.)
a) How many blocks are needed for the relation and its indexes (cf. calculations in Examples 2 and 3, E&N, p. 162, 167)?
b) What is the meaning of the load factor (lower than 100 %)? (answer above!) Should also the load factor of the heap be lower than 100%?
c) Is this kind of ADDRESS index suitable for queries (see the example addresses in the company database, addresses in general)?

2. Assume that we also have a hash based index for the attribute DNO (of the relation EMPLOYEE). Show the structure of this index and calculate the number of pages needed to store a relation given in task 1. (If you need other assumptions do them yourself.)

3. The relation WORKS_ON is implemented as a heap file. In addition, we have a hash-based secondary index to this relation, the index key being ESSN.
a) How this solution relates to the hash file used in task 2.4 (last week)? Is it better or worse? Why?
b) How the relation WORKS_ON should be implemented if we need grouping on projects (PNO) as well as on employees (ESSN)?

4. Assume that relation EMPLOYEE has the indexes given in tasks 1 and 2. Explain all operations (record fetches and record updates; both for the indexes and for the data file) when the following queries are executed:

insert into EMPLOYEE values
('Jim','B','Koch', 223344556, '10-JAN-50', '635 Voss, Houston, TX', 'M', 35000, 123456789, 4);
delete from EMPLOYEE where SSN = '123456789' OR SSN = '333445555';
update EMPLOYEE set SALARY = SALARY * 1.05 where SSN = '123456789';

5. (**) In a B+-tree of order 5 there are 2 to 4 index keys in every node. Assume that the following keys are inserted into an empty B+-tree in that order: 23, 65, 37, 60, 46, 92, 48, 71, 56, 59, 18, 21, 10, 74, 78, 15, 16, 20, and 24.
a) Describe how the B+-tree structure is formed (stepwise). (See E&N, Fig. 6.12; major changes and not all the about 20 steps should be expressed.)
b) How the structure is changed when the keys 24, 23, 10, and 20 are deleted?

6. The indexes of task 1 (for ADDRESS, and for SSN) can be implemented also as a B+ tree index.
a) Show the structure of a leaf node and of a non-leaf node of the ADDRESS index.
b) How many pages are needed for the ADDRESS index?
c) Compare a B+ tree index and an ISAM index for different queries (select.., insert.., delete.., update..). (It is more suitable to consider here the SSN index than the ADDRESS index.)