Exercises

(Course: Database Management Systems)

Chapter 2

Indexing Structures for Files

1. Exercise 18.18 in the text book ("Fundamentals of Database Systems- 6th Edition", Elmasri et al.)

Consider a disk with block size B = 512 bytes. A block pointer is P = 6 bytes long, and a record pointer is PR = 7 bytes long. A file has r = 30,000 **EMPLOYEE** records of *fixed length*. Each record has the following fields: **Name** (30 bytes), **Ssn** (9 bytes), **Department_code** (9 bytes), **Address** (40 bytes), **Phone** (10 bytes), **Birth_date** (8 bytes), **Sex** (1 byte), **Job_code** (4 bytes), and **Salary** (4 bytes, real number). An additional byte is used as a deletion marker.

a. Calculate the record size R in bytes.
b. Calculate the blocking factor *bfr* and the number of file blocks b, assuming an unspanned organization.
c. Suppose that the file is *ordered* by the key field **Ssn** and we want to construct a *primary index* on Ssn. Calculate (i) the index blocking factor bfr_i (which is also the index fan-out fo); (ii) the number of first-level index entries and the number of first-level index blocks; (iii) the number of levels needed if we make it into a multilevel index; (iv) the total number of blocks required by the multilevel index; and (v) the number of block accesses needed to search for and retrieve a record from the file—given its Ssn value—using the primary index.
d. Suppose that the file is *not ordered* by the key field Ssn and we want to construct a *secondary index* on **Ssn**. Repeat the previous exercise (part c) for the secondary index and compare with the primary index.
e. Suppose that the file is *not ordered* by the nonkey field Department_code and we want to construct a *secondary index* on Department_code, using option 3 of Section 18.1.3, with an extra level of indirection that stores record pointers. Assume there are 1,000 distinct values of Department_code and that the EMPLOYEE records are evenly distributed among these values. Calculate (i) the index blocking factor bfr_i (which is also the index fan-out *fo*); (ii) the number of blocks needed by the level of indirection that stores record pointers; (iii) the number of first-level index entries and the number of first-level index blocks; (iv) the number of levels needed if we make it into a multilevel index; (v) the total number of blocks required by the multilevel index and the blocks used in the extra level of indirection; and (vi) the approximate number of block accesses needed to search for and retrieve all records in the file that have a specific Department_code value, using the index.
f. Suppose that the file is *ordered* by the nonkey field Department_code and we want to construct a *clustering index* on Department_code that uses block anchors (every new value of Department_code starts at the beginning of a new block). Assume there are 1,000 distinct values of Department_code and that the EMPLOYEE records are evenly distributed among these values. Calculate (i) the index blocking factor bfr_i (which is also the index fan-out *fo*); (ii) the number of first-level index entries and the number of first-level index blocks; (iii) the number of levels needed if we make it into a multilevel index; (iv) the total number of blocks required by the multilevel index; and (v) the number of block accesses needed to search for and retrieve all records in the file that have a specific Department_code value, using the clustering index (assume that multiple blocks in a cluster are contiguous).

g. Suppose that the file is *not* ordered by the key field Ssn and we want to construct a B+-tree access structure (index) on Ssn. Calculate (i) the orders *p* and *p* leaf of the B+-tree; (ii) the number of leaf-level blocks needed if blocks are approximately 69 percent full (rounded up for convenience); (iii) the number of levels needed if internal nodes are also 69 percent full (rounded up for convenience); (iv) the total number of blocks required by the B+-tree; and (v) the number of block accesses needed to search for and retrieve a record from the file—given its Ssn value—using the B+-tree.

h. Repeat part g, but for a B-tree rather than for a B+-tree. Compare your results for the B-tree and for the B+-tree.

2. Exercise 18.19 in the text book ("Fundamentals of Database Systems- 6th Edition", Elmasri et al.)

PARTS file with Part# as the key field includes records with the following Part# values: 23, 65, 37, 60, 46, 92, 48, 71, 56, 59, 18, 21, 10, 74, 78, 15, 16, 20, 24, 28, 39, 43, 47, 50, 69, 75, 8, 49, 33, 38. Suppose that the search field values are inserted in the given order in a B^+-tree of order p = 4 and p_{leaf} = 3.

Show how the tree will expand and what the final tree will look like.

3. Exercise 18.21 in the text book ("Fundamentals of Database Systems- 6th Edition", Elmasri et al.)

Suppose that the following search field values are deleted, in the given order, from the B^+-tree of Exercise 18.19. The deleted values are 65, 75, 43, 18, 20, 92, 59, 37.

Show how the tree will shrink and show the final tree.