CHAPTER 14: INDEXING STRUCTURES FOR FILES

14.14 Consider a disk with block size B=512 bytes. A block pointer is P=6 bytes long, and a record pointer is P R =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), DEPARTMENTCODE (9 bytes), ADDRESS (40 bytes), PHONE (9 bytes), BIRTHDATE (8 bytes), SEX (1 byte), JOBCODE (4 bytes), 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 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 multi-level index; (iv) the total number of blocks required by the multi-level 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 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 the file is not ordered by the non-key field DEPARTMENTCODE and we want to construct a secondary index on SSN using Option 3 of Section 14.1.3, with an extra level of indirection that stores record pointers. Assume there are 1000 distinct values of DEPARTMENTCODE, 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 a multi-level index; (v) the total number of blocks required by the multi-level 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 having a specific DEPARTMENTCODE value using the index.

(f) Suppose the file is ordered by the non-key field DEPARTMENTCODE and we want to construct a clustering index on DEPARTMENTCODE that uses block anchors (every new value of DEPARTMENTCODE starts at the beginning of a new block). Assume there are 1000 distinct values of DEPARTMENTCODE, 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 a multi-level index; (iv) the total number of blocks required by the multi-level index; and (v) the number of block accesses needed to search for and retrieve all records in the file having a specific DEPARTMENTCODE value using the clustering index (assume that multiple blocks in a cluster are either contiguous or linked by pointers).

(g) Suppose 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% full (rounded up for convenience); (iii) the number of levels needed if internal nodes are also 69% 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.

14.15 A PARTS file with Part# as 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 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 looks like.

14.17 Suppose the following search field values are deleted in the given order from the B + -tree of Exercise 14.15, show how the tree will shrink and show the final tree. The deleted values are: 65, 75, 43, 18, 20, 92, 59, 37.