Lab10 – Indexing Structures

Objective

Indexes are an important part of databases. This assignment helps you to understand the concept of different index types and advantage of using them.

Instructions

To answer the following questions, you may want to read the document ***Ch17 - Examples of the Textbook on Indexes*** available on D2L-> Lecture Slides.

Important: You must explain how you reach to an answer. Writing a number without explanation is not accepted.

- Consider a disk with block size B = 512 bytes.
- A block pointer is P = 6 bytes long, and a record pointer is RP = 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. (A record is composed of fields. A field holds information about an entity. For example, Name and SSN are two fields of an employee's record)

b) Calculate the blocking factor *bfr* and the number of file blocks *b*, assuming an un-spanned 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 one-level primary index and multilevel 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 data file is not ordered by the non-key field Department_code and we want to construct a secondary index on Department_code, using option 3 explained in Appendix A, 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 data file that have a specific Department_code value, using the multilevel index.

f) Suppose that the data file is ordered by the non-key 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
(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 multilevel index (assume that multiple blocks in a cluster are contiguous).

Submission

Submit a PDF file of your answers prior to the due time. Before submission, rename the document as lab10_xy.pdf where x is your first name, and y is your last name.

Marking Guide

Task	Points	Granted	Comment
a	2		
b	3		
c-i	3		
c-ii	5		
c-iii	5		
c-iv	5		
c-v	5		
d-i	3		
d-ii	3		
d-iii	5		
d-iv	5		
d-v	5		
e-i	3		
e-ii	5		
e-iii	5		
e-iv	5		
e-v	5		
e-vi	5		
f-i	3		
f-ii	5		
f-iii	5		
f-iv	5		
f-v	5		
Total	**100**	**0**	

Appendix A

We can create a secondary index on a non-key, non-ordering field of a file. In this case, numerous records in the data file can have the same value for the indexing field. There are several options for implementing such an index:

Option 1 is to include duplicate index entries with the same K(i) value—one for each record. This would be a dense index**.**

Option 2 is to have variable-length records for the index entries, with a repeating field for the pointer. We keep a list of pointers in the index entry for K(i)—one pointer to each block that contains a record whose indexing field value equals K(i). In either option 1 or option 2, the binary search algorithm on the index must be modified appropriately to account for a variable number of index entries per index key value.

Option 3, which is more commonly used, is to keep the index entries themselves at a fixed length and have a single entry for each index field value, but to create an extra level of indirection to handle the multiple pointers. In this non-dense scheme, the pointer P(i) in index entry points to a disk block, which contains a set of record pointers; each record pointer in that disk block points to one of the data file records with value K(i) for the indexing field. If some value K(i) occurs in too many records, so that their record pointers cannot fit in a single disk block, a cluster or linked list of blocks is used.

This technique is illustrated in Figure1. Retrieval via the index requires one or more additional block accesses because of the extra level, but the algorithms for searching the index and (more importantly) for inserting new records in the data file are straightforward. The binary search algorithm is directly applicable to the index file since it is ordered. For range retrievals such as retrieving records where V1 ≤ K ≤ V2, block pointers may be used in the pool of pointers for each value instead of the record pointers. Then a union operation can be used on the pools of block pointers corresponding to the entries from V1 to V2 in the index to eliminate duplicates and the resulting blocks can be accessed. In addition, retrievals on complex selection conditions may be handled by referring to the record pointers from multiple non-key secondary indexes, without having to retrieve many unnecessary records from the data file.

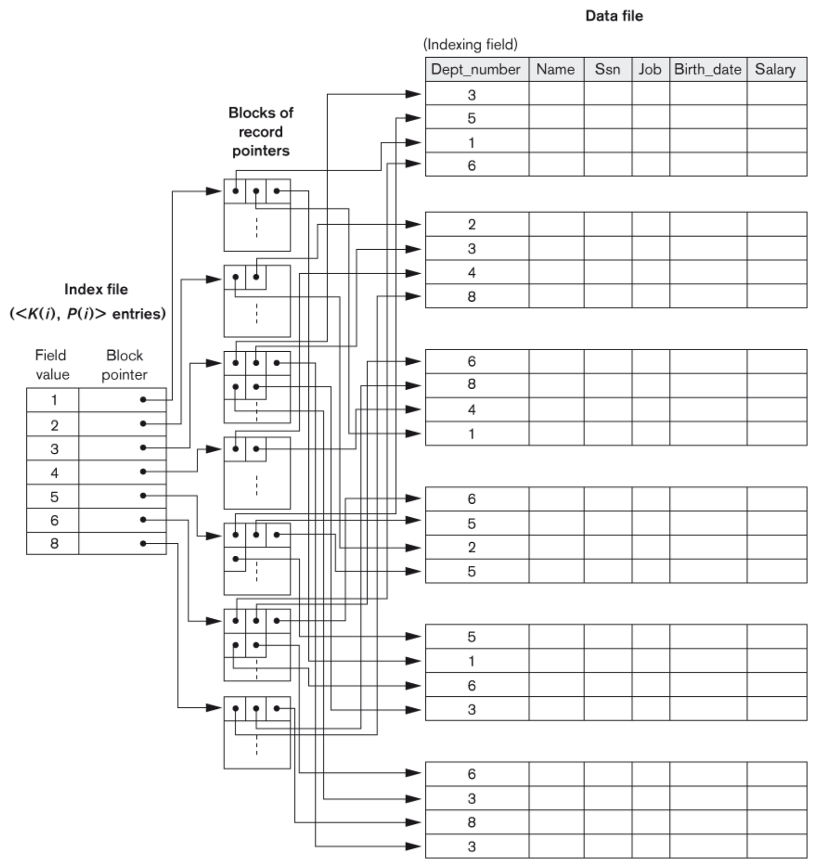

Figure1 - A secondary index (with record pointers) on a non-key field implemented using one level of indirection so that index entries are of fixed length and have unique field values.