Subject: **Database Management Systems** (**CO3021**)
Semester 1, 2018-2019

Chapter 2: Indexing Structures for Files

Question 2.1. What are indexes? Give at least three examples.

Question 2.2. What are primary, secondary, and clustering indexes? Give at least one example for each.

Question 2.3. Compare primary, secondary, and clustering indexes with each other. Which are dense and which are not? Explain the characteristics in their corresponding data file that make them dense or sparse.

Question 2.4. Why can at most one primary or clustering index created on a data file, but zero or many secondary indexes? Give an example to demonstrate your answer.

Question 2.5. Distinguish between single-level indexes and multilevel indexes. Give an example to demonstrate your answer.

Question 2.6. Describe B-tree and B+-tree when they are used as secondary access structures for a data file. Distinguish between B-tree and B+-tree. Give an example for each structure.

NOTE: The following exercises are extracted from those in chapter 17 of [1]: *R. Elmasri, S. R. Navathe, Fundamentals of Database Systems- 7th Edition, Pearson, 2016.*

Question 2.7. What is partitioned hashing? How does it work? What are its limitations?

Question 2.8. What is a grid file? What are its advantages and disadvantages?

Question 2.9. What is a fully inverted file? What is an indexed sequential file?

Question 2.10. How can hashing be used to construct an index?

Question 2.11. What is bitmap indexing? Create a relation with two columns and sixteen tuples and show an example of a bitmap index on one or both.

Question 2.12. 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 *bfri* (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 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***, 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 *bfri* (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;

(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 *bfri* (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 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 *pleaf* 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;

(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.

Question 2.13. A 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.

Question 2.14. Repeat **Question 2.13**, but use a **B-tree of order *p* = 4** instead of a **B+-tree**.

Question 2.15. Suppose that the following search field values are deleted, in the given order, from the B+-tree of **Question 2.13**; show how the tree will shrink and show the final tree. The deleted values are 65, 75, 43, 18, 20, 92, 59, 37.

Question 2.16. Repeat **Question 2.15**, but for the **B-tree** of **Question 2.14**.

Question 2.17. Suppose that several secondary indexes exist on nonkey fields of a file, implemented using ***option 3***; for example, we could have secondary indexes on the fields Department_code, Job_code, and Salary of the EMPLOYEE file of **Question 2.12**. Describe an efficient way to search for and retrieve records satisfying a complex selection condition on these fields, such as (Department_code = 5 AND Job_code = 12 AND Salary = 50,000), using the record pointers in the indirection level.