CS3223: Database Management Systems

Tutorial 3

(Week 5, Feb 2022)

1. (Exercise 13.3 from main text) Suppose that you just finished inserting several records into a heap file and now want to sort those records. Assume that the DBMS uses external sort and makes efficient use of the available buffer space when it sorts a file. Here is some potentially useful information about the newly loaded file and the DBMS software available to operate on it: The number of records in the file is 4500. The sort key for the file is 4 bytes long. You can assume that pointers are 8 bytes long. Each record is a total of 48 bytes long. The page size is 512 bytes. Each page has 12 bytes of control information on it. Four buffer pages are available.
    a. How many sorted subfiles will there be after the initial pass of the sort, and how long will each subfile be?
    b. How many passes (including the initial pass just considered) are required to sort this file?
    c. What is the total I/O cost for sorting this file?
    d. Suppose that you have a B+ tree index with the search key being the same as the desired sort key. Find the cost of using the index to retrieve the records in sorted order for each of the following cases:
        i. The index uses Alternative (1) for data entries.
        ii. The index uses Alternative (2) and is unclustered. (You can compute the worst-case cost in this case.)

2. Consider a disk with an average seek time of 10ms, average rotational delay of 5ms and a transfer time of 1ms for a 4K page. Assume that the cost of reading/writing a page is the sum of these values (i.e., 16 ms) unless a sequence of pages is read/written. In this case, the cost is the average seek plus the average rotational delay (to find the first page in the sequence) plus 1 ms per page (to transfer data). You are given 320 buffer pages and asked to sort a file with 10,000,000 pages. Assume that you begin by creating sorted runs of 320 pages each in the first pass. Evaluate the cost of the following approaches for the subsequent merging passes:

    a) Do 319-way merges. If there are fewer than 319 runs to merge, then allocate one input buffer per run, and one output buffer, i.e., ignore the other "spare" buffer pages.
    b) Create 256 'input' buffers of 1 page each, create an 'output' buffer of 64 pages, and do 256-way merges.
    c) Create 16 'input' buffer of 16 pages each, create an 'output' buffer of 64 pages, and do 16-way merges.
    d) Create 4 'input' buffer of 64 pages each, create an 'output' buffer of 64 pages, and do 4-way merges.

    What conclusion(s) can you draw from these results?

    [For simplicity, you can ignore the effect of a final read/written block that is slightly smaller than the earlier blocks, i.e., treat that as the same as other blocks.]

3. Suppose we have a relation whose n tuples each require R bytes, and we have a machine whose main memory M bytes and disk-block size B are just sufficient to sort the n tuples in 2 passes (i.e., first pass to generate runs, and second pass to merge the runs). How would the maximum n change if we change one of the parameters as follows?

   a) Double B
   b) Double R
   c) Double M

Consider the refinement to the external sort algorithm that produces runs of length 2B on average, where B is the number of buffer pages (this is the replacement selection algorithm). This refinement was described in Section 11.2.1 under the assumption that **all records are the same size**. Explain why this assumption is required and extend the idea to cover the case of variable length records.

The assumption that all records are of the same size is used when the algorithm moves the smallest entry with a key value larger than k to the output buffer and replaces it with a value from the input buffer.
This "replacement" will only work if the records are of the same size. If the entries are variable size, then we must also keep track of the size of each entry, and replace the moved entry with a new entry that fits in the available memory location