Sample Exercises: External Sorting

Exercise 1 Answer the following questions for each of these scenarios, assuming that our most general external sorting algorithm is used:

(a) A file with 10,000 pages and three available buffer pages.

(b) A file with 20,000 pages and five available buffer pages.

(c) A file with 2,000,000 pages and 17 available buffer pages.

1. How many runs will you produce in the first pass?
2. How many passes will it take to sort the file completely?
3. What is the total I/O cost of sorting the file?
4. How many buffer pages do you need to sort the file completely in just two passes?

Exercise 2 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 rids are 8 bytes long and page ids are 4 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.

1. How many sorted subfiles will there be after the initial pass of the sort, and how long will each subfile be?
2. How many passes (including the initial pass just considered) are required to sort this file?
3. What is the total I/O cost for sorting this file?
4. What is the largest file, in terms of the number of records, you can sort with just four buffer pages in two passes? How would your answer change if you had 257 buffer pages?

5. 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:

   - The index uses Alternative (1) for data entries.
   - The index uses Alternative (2) and is unclustered. (You can compute the worst-case cost in this case.)
   - How would the costs of using the index change if the file is the largest that you can sort in two passes of external sort with 257 buffer pages? Give your answer for both clustered and unclustered indexes.

Solutions

Answer 1 The answer to each question is given below.

1. In the first pass (Pass 0), $\lceil N/B \rceil$ runs of B pages each are produced, where N is the number of file pages and B is the number of available buffer pages:

(a) $\lceil 10000/3 \rceil = 3334$ sorted runs.

(b) $\lceil 20000/5 \rceil = 4000$ sorted runs.

(c) $\lceil 2000000/17 \rceil = 117648$ sorted runs.

2. The number of passes required to sort the file completely, including the initial sorting pass, is $\lceil \log_{B-1} N1 \rceil + 1$, where $N1 = \lceil N/B \rceil$ is the number of runs produced by Pass 0:

(a) $\lceil \log_2 3334 \rceil + 1 = 13$ passes.

(b) $\lceil \log_4 4000 \rceil + 1 = 7$ passes.

(c) $\lceil \log_{16} 117648 \rceil + 1 = 6$ passes.

3. Since each page is read and written once per pass, the total number of page I/Os for sorting the file is $2 * N * (\#passes)$:

(a) 2*10000*13 = 260000.

(b) 2*20000*7 = 280000.

(c) 2*2000000*6 = 24000000.

4. In Pass 0, $\lceil N/B \rceil$ runs are produced. In Pass 1, we must be able to merge this many runs; i.e., $B - 1 \geq \lceil N/B \rceil$. This implies that B must at least be large enough to satisfy $B * (B - 1) \geq N$; this can be used to guess at B, and the guess must be validated by checking the first inequality. Thus:

(a) With 10000 pages in the file, B = 101 satisfies both inequalities, B = 100 does not, so we need 101 buffer pages.

(b) With 20000 pages in the file, B = 142 satisfies both inequalities, B = 141 does not, so we need 142 buffer pages.

(c) With 2000000 pages in the file, B = 1415 satisfies both inequalities, B = 1414 does not, so we need 1415 buffer pages.

Answer 2 The answer to each question is given below.

1. Assuming that the general external merge-sort algorithm is used, and that the available space for storing records in each page is 512−12 = 500 bytes, each page can store up to 10 records of 48 bytes each. So, 450 pages are needed in order to

store all 4500 records, assuming that a record is not allowed to span more than one page. Given that 4 buffer pages are available, there will be $\lceil 450/4 \rceil = 113$ sorted runs (sub-files) of 4 pages each, except the last run, which is only 2 pages long.

2. The total number of passes will be equal to $\log_3 113 + 1 = 6$ passes.

3. The total I/O cost for sorting this file is $2 * 450 * 6 = 5400$ I/Os.

4. As we saw in the previous exercise, in Pass 0, $\lceil N/B \rceil$ runs are produced. In Pass 1, we must be able to merge this many runs; i.e., $B - 1 \geq \lceil N/B \rceil$. When B is given to be 4, we get N = 12. The maximum number of records on 12 pages is $12 * 10 = 120$. When B = 257, we get N = 65792, and the number of records is $65792 * 10 = 657920$.

5.

    a. If the index uses Alternative (1) for data entries, and it is clustered, the cost will be equal to the cost of traversing the tree from the root to the leftmost leaf plus the cost of retrieving the pages in the sequence set. Assuming 67% occupancy, the number of leaf pages in the tree (the sequence set) is 450/0.67 = 600.

    b. If the index uses Alternative (2), and is not clustered, in the worst case, first we scan B+ tree's leaf pages, also each data entry will require fetching a data page. The number of data entries is equal to the number of data records, which is 4500. Since there is one data entry per record, each data entry requires 12 bytes, and each page holds 512 bytes, the number of B+ tree leaf pages is about $(4500 * 12)/(512 * 0.67))$, assuming 67% occupancy, which is about 150. Thus, about 4650 I/Os are required in a worst-case scenario.

c. The B+ tree in this case has 65792/0.67 = 98197 leaf pages if Alternative (1) is used, assuming 67% occupancy. This is the number of I/Os required (plus the relatively minor cost of going from the root to the left-most leaf). If Alternative (2) is used, and the index is not clustered, the number of I/Os is approximately equal to the number of data entries in the worst case, that is 657920, plus the number of B+ tree leaf pages. The number of the B+ leaf pages is now calculated as follows. We index 657920 entries, for each entry we have the sort key (4b) and the row id (8b) so 12 bytes. The page size is 512b. Overall #leaf pages: (657920 entries * 12b/entry)/(512b/page * 0.67) = 23015. Total IOs: 657920+23015=680935.