18.2 Multilevel Indexes

The indexing schemes we have described thus far involve an ordered index file. A binary search is applied to the index to locate pointers to a disk block or to a record (or records) in the file having a specific index field value. A binary search requires approximately ($\log_2 b_i$) block accesses for an index with b_i blocks because each step of the algorithm reduces the part of the index file that we continue to search by a factor of 2. This is why we take the log function to the base 2. The idea behind a **multilevel index** is to reduce the part of the index that we continue to search by bfr_i, the blocking factor for the index, which is larger than 2. Hence, the search space is reduced much faster. The value bfr_i is called the **fan-out** of the multilevel index, and we will refer to it by the symbol ***fo***. Whereas we divide the *record search space* into two halves at each step during a binary search, we divide it n-ways (where n = the fan-out) at each search step using the multilevel index. Searching a multilevel index requires approximately ($\log_{fo} b_i$) block accesses, which is a substantially smaller number than for a binary search if the fan-out is larger than 2. In most cases, the fan-out is much larger than 2.

A multilevel index considers the index file, which we will now refer to as the **first** (or **base**) **level** of a multilevel index, as an *ordered file* with a *distinct value* for each $K(i)$. Therefore, by considering the first-level index file as a sorted data file, we can create a primary index for the first level; this index to the first level is called the **second level** of the multilevel index. Because the second level is a primary index, we can use block anchors so that the second level has one entry for *each block* of the first level. The blocking factor bfr_i for the second level—and for all subsequent levels—is the same as that for the first-level index because all index entries are the same size; each has one field value and one block address. If the first level has r_1 entries, and the blocking factor—which is also the fan-out—for the index is $bfr_i = fo$, then the first level needs $\lceil (r_1/fo) \rceil$ blocks, which is therefore the number of entries r_2 needed at the second level of the index.

We can repeat this process for the second level. The **third level**, which is a primary index for the second level, has an entry for each second-level block, so the number of third-level entries is $r_3 = \lceil (r_2/fo) \rceil$. Notice that we require a second level only if the first level needs more than one block of disk storage, and, similarly, we require a third level only if the second level needs more than one block. We can repeat the preceding process until all the entries of some index level t fit in a single block. This block at the tth level is called the **top** index level.[4] Each level reduces the number of entries at the previous level by a factor of fo—the index fan-out—so we can use the formula $1 \leq (r_1/((fo)^t))$ to calculate t. Hence, a multilevel index with r_1 first-level entries will have approximately t levels, where $t = \lceil (\log_{fo}(r_1)) \rceil$. When searching the

[4]The numbering scheme for index levels used here is the reverse of the way levels are commonly defined for tree data structures. In tree data structures, t is referred to as level 0 (zero), $t - 1$ is level 1, and so on.