EX1: Search Cost with Index versus Binary Search

Example 1. Suppose that we have an ordered file with $r = 300{,}000$ records stored on a disk with block size $B = 4{,}096$ bytes.[5] File records are of fixed size and are unspanned, with record length $R = 100$ bytes. The blocking factor for the file would be $bfr = \lfloor (B/R) \rfloor = \lfloor (4{,}096/100) \rfloor = 40$ records per block. The number of blocks needed for the file is $b = \lceil (r/bfr) \rceil = \lceil (300{,}000/40) \rceil = 7{,}500$ blocks. A binary search on the data file would need approximately $\lceil \log_2 b \rceil = \lceil (\log_2 7{,}500) \rceil = 13$ block accesses.

Now suppose that the ordering key field of the file is $V = 9$ bytes long, a block pointer is $P = 6$ bytes long, and we have constructed a primary index for the file. The size of each index entry is $R_i = (9 + 6) = 15$ bytes, so the blocking factor for the index is $bfr_i = \lfloor (B/R_i) \rfloor = \lfloor (4{,}096/15) \rfloor = 273$ entries per block. The total number of index entries r_i is equal to the number of blocks in the data file, which is 7,500. The number of index blocks is hence $b_i = \lceil (r_i/bfr_i) \rceil = \lceil (7{,}500/273) \rceil = 28$ blocks. To perform a binary search on the index file would need $\lceil (\log_2 b_i) \rceil = \lceil (\log_2 28) \rceil = 5$ block accesses. To search for a record using the index, we need one additional block access to the data file for a total of $5 + 1 = 6$ block accesses—an improvement over binary search on the data file, which required 13 disk block accesses. Note that the index with 7,500 entries of 15 bytes each is rather small (112,500 or 112.5 Kbytes) and would typically be kept in main memory thus requiring negligible time to search with binary search. In that case we simply make one block access to retrieve the record.

EX2: Clustering Index Cost

Example 2. Suppose that we consider the same ordered file with $r = 300{,}000$ records stored on a disk with block size $B = 4{,}096$ bytes. Imagine that it is ordered by the attribute Zipcode and there are 1,000 zip codes in the file (with an average 300 records per zip code, assuming even distribution across zip codes.) The index in this case has 1,000 index entries of 11 bytes each (5-byte Zipcode and 6-byte block pointer) with a blocking factor $bfr_i = \lfloor(B/R_i)\rfloor = \lfloor(4{,}096/11)\rfloor = 372$ index entries per block. The number of index blocks is hence $b_i = \lceil(r_i/bfr_i)\rceil = \lceil(1{,}000/372)\rceil = 3$ blocks. To perform a binary search on the index file would need $\lceil(\log_2 b_i)\rceil = \lceil(\log_2 3)\rceil = 2$ block accesses. Again, this index would typically be loaded in main memory (occupies 11,000 or 11 Kbytes) and takes negligible time to search in memory. One block access to the data file would lead to the first record with a given zip code.

EX3: Search Cost with Secondary Index

Example 3. Consider the file of Example 1 with $r = 300{,}000$ fixed-length records of size $R = 100$ bytes stored on a disk with block size $B = 4{,}096$ bytes. The file has $b = 7{,}500$ blocks, as calculated in Example 1. Suppose we want to search for a record with a specific value for the secondary key—a nonordering key field of the file that is $V = 9$ bytes long. Without the secondary index, to do a linear search on the file would require $b/2 = 7{,}500/2 = 3{,}750$ block accesses on the average. Suppose that we construct a secondary index on that *nonordering key* field of the file. As in Example 1, a block pointer is $P = 6$ bytes long, so each index entry is $R_i = (9 + 6) = 15$ bytes, and the blocking factor for the index is $bfr_i = \lfloor (B/R_i) \rfloor = \lfloor (4{,}096/15) \rfloor = 273$ index entries per block. In a dense secondary index such as this, the total number of index entries r_i is equal to the *number of records* in the data file, which is 300,000. The number of blocks needed for the index is hence $b_i = \lceil (r_i/bfr_i) \rceil = \lceil (300{,}000/273) \rceil = 1{,}099$ blocks.

EX4: Search Cost with Multilevel Index

Example 4. Suppose that the dense secondary index of Example 3 is converted into a multilevel index. We calculated the index blocking factor $bfr_i = 273$ index entries per block, which is also the fan-out *fo* for the multilevel index; the number of first-level blocks $b_1 = 1{,}099$ blocks was also calculated. The number of second-level blocks will be $b_2 = \lceil (b_1/fo) \rceil = \lceil (1{,}099/273) \rceil = 5$ blocks, and the number of third-level blocks will be $b_3 = \lceil (b_2/fo) \rceil = \lceil (5/273) \rceil = 1$ block. Hence, the third level is the top level of the index, and $t = 3$. To access a record by searching the multilevel index, we must access one block at each level plus one block from the data file, so we need $t + 1 = 3 + 1 = 4$ block accesses. Compare this to Example 3, where 12 block accesses were needed when a single-level index and binary search were used.