(these are known as *variable-length records*). The number of fields in the record minus one (the field 1 with the descriptor name) indicates the number of documents indexed by the descriptor in field 1. This number is the frequency of occurrence of the given descriptor in the file. Usually, this frequency is also stored in the file, yet in this case (to simplify things) it is not indicated. Also notice that the documents' numbers in the record are arranged in order of their arrival into the system, that is, in ascending order of numbers. Now that we have considered an inverted file, we will proceed to the main point and see how a subject search is carried out in file organization.

Let us assume that as a result of indexing a search request, the following query formulation has been constructed.

A2 OR (A34 AND B7) OR (D93 AND E16 AND F56)

The search procedure will be demonstrated using this query formulation consisting of three subrequests with one, two, and three descriptors, respectively. A search is performed for each of these subrequests, and it could be done in any order. The simplest search is for the first subrequest, that is, subrequest A2 consisting of one descriptor. In this case, the record containing descriptor A2 is retrieved from the 800 records. The retrieved record (with A2 in field 1) is one of the records presented in Figure 8.6. In it there are four fields with the numbers of the documents that have been indexed by descriptor A2. It means that at the stage of search for the first subrequest, four formally relevant documents have been found, and the numbers (addresses) corresponding to them are held in primary storage. This is followed by a search for the second subrequest consisting of two descriptors: A34 and B7. In this case two records are retrieved, corresponding to the given subrequest descriptors. Then the numbers, written in these records, are compared and those that are contained in every record (the same numbers) are again selected and put into the system's memory, since the documents corresponding to these numbers are considered formally relevant. Let us illustrate this process with the following example (Figure 8.7).

Let us suppose that descriptor A34 appears in 17 documents (more precisely, document profiles) and descriptor B7 appears in 46 documents. Then the record with descriptor A34 will have 17 numbers and the record with descriptor B7 will have 46 numbers. The comparison of every number from record A34 with the numbers in record B7 revealed that only 2 numbers, namely 312 and 3677, are common to both records. Hence, the descriptors A34 and B7 are simultaneously contained in only 2 documents (numbered 312 and 3677). More formally, this process (the process of search for the second subrequest) can be presented as follows. Let S denote the set of the document numbers available in the system. In our example $|S| = 10{,}000$. Symbol A will denote the set of document numbers represented in record A34, and symbol B will denote the set of document numbers in record B7. It is quite evident that $A \subseteq S$ and $B \subseteq S$, with $|A| = 17$, and $|B| = 46$. The result of the search process (let it be N) is the intersection of sets A and B; that is, $N = A \cap B = \{312, 3677\}$.

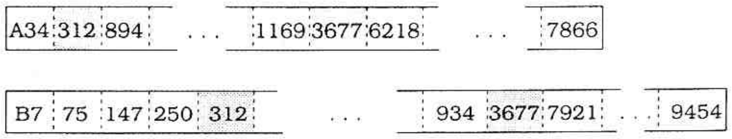

Figure 8.7
An example of a search for subrequests consisting of two descriptors.

Let us now consider a subject search for the subrequest consisting of descriptors D93, E16, and F56. In this case there are only 3 records, of the available 800, corresponding to the indicated descriptors. Then the numbers from the shortest record (of the 3 found records) are compared with the numbers of the next longest record. As a result of such a comparison, which incidentally is analogous to the comparison for the second subrequest, all numbers, which are contained in both compared records, are selected. Suppose we compared record D93 with record E16 and discovered 9 numbers contained in both records. Then the found numbers are compared with record F56. Let us also assume that 2 of the 9 are also contained in record F56. This means that a search for the third subrequest found 2 numbers (addresses) of formally relevant documents. Now, if the set of numbers from record D93 is denoted by D, the set of numbers from record E16 by E, and the set of numbers from record F56 by F, then $N = (D \text{ AND } E) \text{ AND } F$ and $|N| = 2$. Because there were only three subrequests in the query formulation used as an example, after the search for the third subrequest a subject search for the input request is considered over. Then comes the next stage, that of an address search.

An address search begins by comparing all the numbers, found in the subject search with the aim of finding duplicates. After eliminating duplicates, the address of every document in the file of documents is calculated (for example, by using a hashing algorithm), and the output for the user is formed.

We have now considered the search process with the use of an inverted file and can now note its main advantage. The use of an inverted file allows us to substantially reduce (often, by orders of magnitude) search time and, consequently, to substantially improve the system's performance. Indeed, returning to our example we may now note that sequential access required comparison of query formulation with 10,000 document profiles written in 10,000 records. In the case of inverted file organization, however, the search will need only 6 of the 800 records available in the system (because the given query formulation contains only 6 descriptors). This leads to the conclusion that the use of inverted file organization cuts the number of comparisons (in our case 100 times less), reduces the number of I/O operations, and improves the time of search. This is why inverted files are used in most functioning IR systems.