Inverted indexes are the most fundamental and widely used data structures in information retrieval. For each unique word occurring in a document collection, the inverted index stores a list of the documents in which this word occurs. Compression techniques are often applied to further reduce the space requirement of these lists.

To process a phrase query, you still need to access the inverted index entries for each distinct term. As before, you would start with the least frequent term and then work to further restrict the list of possible candidates. In the merge operation, the same general technique is used as before, but rather than simply checking that both terms are in a document, you also need to check that their positions of appearance in the document are compatible with the phrase query being evaluated. This requires working out offsets between the words.

Worked example. Satisfying phrase queries.phrasequery Suppose the postings lists for to and be are as in Figure 2.11 , and the query is ``to be or not to be''. The postings lists to access are: to, be, or, not. We will examine intersecting the postings lists for to and be. We first look for documents that contain both terms. Then, we look for places in the lists where there is an occurrence of be with a token index one higher than a position of to, and then we look for another occurrence of each word with token index 4 higher than the first occurrence. In the above lists, the pattern of occurrences that is a possible match is:

Different approaches have been proposed to support phrase searching using an inverted index. One strategy is to maintain the position information in the inverted list. That is, for each document d in the inverted list of a word w, we store the positions at which w occurs in d. The positions corresponding to each d in the list can be sorted so as to achieve compression (using encoding functions like gap, gamma, or delta) [15].