Full tekst serach

Preliminary information: examples are based on a database
sent in a folder

As the name suggests, today we will mainly deal with databases with the following data
and only such will be of interest to us.
Let's start with a short introduction, probably each of us knows the function of LIKE, is used to
finding a given word in WHERE clause, this word may be preceded by any of the following
the number of characters and I can occur in this word, but let's ask ourselves if it is
Is the inquiry effective? No, we do not use LIKE to browse each line of data.
looking for a specific pattern, so as not difficult to deduce in the database which
has hundreds of thousands of poems, such a search may take moments of time.
What are indexes? Simply put, we can compare them to the table of contents in the book.
So looking for a specific element we no longer need to browse the entire table because
By building an index on a searched table, our search is much faster.
So we can ask ourselves whether we should create indexes on each table in order to
speed up possible search results? No, the created index takes away from us
disk space so it is recommended to use indexes only on tables often
searched.
So what is a full-text search? This search allows us to search for
textual data recorded in tables and what can sometimes be helpful to the FTS may be
Use to search binary files. So what can you do with FTS?
We can replace the LIKE function with much faster queries.
Functionality. We can search for a few words, different grammatical forms of given words, words
synonyms and synonyms. We can also look for separated words with a specific
number of other words, you can also pay attention to the search words / phrases so
that they remain with us.
returned most accurate searches along with their weight/ranking.
Now I will discuss how to create such indexes, at the very beginning we have to create a directory
grouping full-text indices, then you should create a unique index for each of them
column in the table that we want to customize for full-text search. Indexes

full-text do not have their name because in one table there can be only one such one.
index. It can be created on one or more columns. Such indexes can be
switch on and off. A complete example of how to do this can be found in the Examples file
sql.txt. To use the search we will use two functions used
in clause WHERE: freetext and contains. In the file with examples I have placed a lot of
examples of how these functions work. The general operation of freetext is such that without any
additional restrictions are always searched for by all forms of the word in question.
with synonyms, but we cannot use logical operators, prefixes, we do not have
We cannot search for words that are close to each other. Contains for
it returns to us in its basic form only the occurrence of a given word, but accordingly
by modifying the data, the query can also find us various flexural forms, synonyms,
can use prefixes, words have their own weights, operators can be used
logical. In summary, Contains is a much quicker and more
precise, returning a smaller number of results but at the same time this is a query.
more complex than freetext. In the examples I created tables in which I placed
synonyms of two words, freetext query will return the whole table to me when I search
any word contained in this table and the usual contains will return only one of them
the word you are looking for.
I'll briefly tell you more about phrasing for the phrases you're looking for, using the function
Freetextable we put it in the clause from then in brackets specified tables where it has
search, columns to be searched and word/phrase to be found, results
are returned only in the form of the id of the given line and its ranking, so that it looks like this
normally we have to slightly modify our query, the example contained in the file.
Let us build such a query:
Select * from sys.dm_fts_parser('"A kitten put on a fence and blinks, it's a nice song not
long', 1045, 0,0)
where 1045 means Polish language by 0 means that we are using the system.
vocabulary of search terms.
After compiling, we can see that some of the words of type i, in the special_term table, are defined by
as noise word, these are the so-called noisy words, words that are omitted, not taken under
Attention when counting the closeness of words and their weight. These words are defined by SQL
Server, we can check them (shown in the preview for Polish and English),
We can also define such a list ourselves, why do we need to do it? For example, by browsing through a list of
documentation of the program in Java often you can find there the word array, which

will not be for
so we add it to our list. Example included in the file with examples.
We will now discuss the thesaurus dictionary, it's an ordinary xml file that I find in this file.
location: C:➤Program Files【Microsoft SQL ServerMLINESQL13.SQLEXPRESSESS_SQLQLQLQLFTData
we have there files available for many languages, we are interested in tsplk file → dictionary for the language
Polish. Thesaurus is a dictionary of synonyms that we can define ourselves, freetext always uses this a contains with an appropriately modified query.
By default each file is commented, let's look at the lines I changed:

```
<expansion>
 <sub>pole</sub>
 <sub>dwor</sub>
 <sub>zewnatrz</sub>
 </expansion>
```

We mean that if someone searches for the word field, he will also search for the manor house and
Outside, queries can be permutated as desired. We enter our own synonym there, people from Cracow go to the field, from Warsaw to the manor house and still others can go to the following places
external walk it all means in fact the same.
Another clause:

```
<replacement>
 <pat>Konwersatorium</pat>
 <sub>Wyklad</sub>
 </replacement>
```

While searching for the word "Conversion", we will find the word "Lecture, but
As you can see, we are replacing the search for one of the following
second words. After making these changes, the file should be uncommented, the dictionary should be reloaded.
in the sql server and work. Some examples I've included in the file with examples.
I mentioned about binarncyh files already at the beginning so there is no need to repeat, we can
on them to do exactly the same as on the text data in the tables.
Thank you for listening and I hope that the knowledge I have presented will be useful to you.
in working with SQL Server.

Sławomir Guzik