

Re: why does MATCH/AGAINST fail to catch entries that LIKE does catch?

## Re: why does MATCH/AGAINST fail to catch entries that LIKE does catch?

---

*Source:* <http://coding.derkeiler.com/Archive/PHP/comp.lang.php/2007-06/msg01302.html>

---

- *From:* Rik <luiheidsgoeroe@xxxxxxxxxxxx>
  - *Date:* Tue, 19 Jun 2007 10:34:29 +0200
- 

On Tue, 19 Jun 2007 09:08:18 +0200, lawrence k <lkrubner@xxxxxxxxxxxx> wrote:

Wierd. Go to this page:

<http://www.ihanuman.com/search.php>

and search for "yoga"

This query gets run:

```
SELECT * FROM albums WHERE MATCH(name,description) AGAINST ('yoga')  
ORDER BY id DESC
```

it returns nothing. (other searches work, but not the one for "yoga").

But if I do `SELECT * FROM albums WHERE description LIKE '%yoga%'`

then I get 5 matches

This clearly a MySQL issue, NOT PHP.

However, I think this might shed some light, from the manual:

"The MySQL FULLTEXT implementation regards any sequence of true word characters (letters, digits, and underscores) as a word. That sequence may also contain apostrophes ('?'), but not more than one in a row. This means that `aaa'bbb` is regarded as one word, but `aaa"bbb` is regarded as two words. Apostrophes at the beginning or the end of a word are stripped by the FULLTEXT parser; `'aaa'bbb'` would be parsed as `aaa'bbb`.

The FULLTEXT parser determines where words start and end by looking for certain delimiter characters; for example, `? ?` (space), `?,?` (comma), and `?.` (period). If words are not separated by delimiters (as in, for example, Chinese), the FULLTEXT parser cannot determine where a word begins or ends. To be able to add words or other indexed terms in such languages to a FULLTEXT index, you must preprocess them so that they are separated by some arbitrary delimiter such as `?".?`

Some words are ignored in full-text searches:

Re: why does MATCH/AGAINST fail to catch entries that LIKE does catch?

## Re: why does MATCH/AGAINST fail to catch entries that LIKE does catch?

Any word that is too short is ignored. The default minimum length of words that are found by full-text searches is four characters.

Words in the stopword list are ignored. A stopword is a word such as 'the' or 'some' that is so common that it is considered to have zero semantic value. There is a built-in stopword list, but it can be overwritten by a user-defined list."

"Every correct word in the collection and in the query is weighted according to its significance in the collection or query. Consequently, a word that is present in many documents has a lower weight (and may even have a zero weight), because it has lower semantic value in this particular collection. Conversely, if the word is rare, it receives a higher weight. The weights of the words are combined to compute the relevance of the row.

Such a technique works best with large collections (in fact, it was carefully tuned this way). For very small tables, word distribution does not adequately reflect their semantic value, and this model may sometimes produce bizarre results. For example, although the word 'MySQL' is present in every row of the articles table shown earlier, a search for the word produces no results:

```
mysql> SELECT * FROM articles
-> WHERE MATCH (title,body) AGAINST ('MySQL');
Empty set (0.00 sec)
```

The search result is empty because the word 'MySQL' is present in at least 50% of the rows. As such, it is effectively treated as a stopword. For large datasets, this is the most desirable behavior: A natural language query should not return every second row from a 1GB table. For small datasets, it may be less desirable."

So, in short: 'yoga' might not be found as a separate word, or be considered to 'common' to match. For more details, ask a MySQL-group.

—  
Rik Wasmus

.