PostgreSQL Full Text Search and Trigram Confusion

Asked 11 years, 8 months ago Modified 1 year, 2 months ago Viewed 14k times

52

I'm a little bit confused with the whole concept of PostgreSQL, full text search and Trigram. In my full text search queries, I'm using tsvectors, like so:

```
SELECT * FROM articles
WHERE search_vector @@ plainto_tsquery('english', 'cat, bat, rat');
```

The problem is, this method doesn't account for misspelling. Then I started to read about Trigram and `pg_trgm`:

Looking through other examples, it seems like trigram is used or vectors are used, but never both. So my questions are: Are they ever used together? If so, how? Does trigram replace full text? Are trigrams more accurate? And how are trigrams on performance?

postgresql full-text-search pattern-matching

Share
Improve this question
Follow

edited Apr 8, 2013 at 18:55
Erwin Brandstetter
654k ● 156 ● 1.1k ● 1.3k

asked Apr 8, 2013 at 16:30
Devin Dixon
12.3k ● 24 ● 97 ● 176

1 Answer

Sorted by: Highest score (default)

80

They serve very different purposes.

- Full Text Search is used to return documents that match a search query of stemmed words.
- Trigrams give you a method for comparing two strings and determining how similar they look.

Consider the following examples:

```
SELECT 'cat' % 'cats'; --true
```

The above returns true because `'cat'` is quite similar to `'cats'` (as dictated by the pg_trgm limit).

```
SELECT 'there is a cat with a dog' % 'cats'; --false
```

The above returns `false` because `%` is looking for similarity between the two entire strings, not looking for the word `cats` *within* the string.

```
SELECT to_tsvector('there is a cat with a dog') @@ to_tsquery('cats'); --true
```

This returns `true` because tsvector transformed the string into a list of stemmed words and ignored a bunch of common words (stop words - like 'is' & 'a')... then searched for the stemmed version of `cats`.

It sounds like you want to use trigrams to **auto-correct** your `ts_query` but that is not really possible (not in any efficient way anyway). They do not really *know* a word is misspelt, just how similar it might be to another word. They *could* be used to search a table of words to try and find similar words, allowing you to implement a "did you mean..." type feature, but this word require maintaining a separate table containing all the words used in your `search` field.

If you have some commonly misspelt words/phrases that you want the text-index to match you might want to look at Synonym Dictorionaries

Share

Improve this answer

Follow

edited Oct 10, 2023 at 8:38
Jan Klimo
4,920 ● 2 ● 38 ● 44

answered Apr 8, 2013 at 17:01
Chris Farmiloe
14.2k ● 5 ● 49 ● 57

2 I've added a couple of examples to highlight the differences between `%` and `@@` from each extension. If your aim is to find documents that contain english (or any known language that you have a dictionary for) then you are after full-text. If your aim is to match an entire field against a string of the entire field with a bit of leeweigh for typos, then pg_trgm is what you want. – Chris Farmiloe Apr 8, 2013 at 18:16

Thanks for the explanation! That cleared it up a lot. Ok so it looks like the problem can be solved by expanding my knowledge of dictionaries. – Devin Dixon Apr 8, 2013 at 18:18

20 I love this answer. I hate the fact that this answer is accurate :(– courtsimas Jul 3, 2015 at 22:26

8 The trigram module (pg_trgm) now has "word similarity" functionality since Postgres 9.6 - i.e. it can look for the most similar word inside the string, rather than comparing the query against the string in its entirety. – Inkling Jun 24, 2017 at 14:52