

Full Text Search in PostgreSQL

Aleksander Alekseev

Agenda

- Intro
- Full text search basics
- Fuzzy full text search
- And some other topics

Intro



PROFESSIONAL
Posigres

Well-known FTS Solutions

- ElasticSearch
- Solr
- Sphinx

Why Use FTS in PostgreSQL

- More or less as good as specialized software
- No data duplication
- Data is always consistent
- No need to install and maintain anything except PostgreSQL

Full Text Search Basics

to_tsvector

```
# SELECT to_tsvector('No need to install and maintain anything except PostgreSQL');  
  
'anyth':7 'except':8 'instal':4 'maintain':6 'need':2 'postgresql':9  
  
(1 row)
```

```
# SELECT to_tsvector('russian',  
  
    'Не нужно устанавливать и поддерживать ничего кроме PostgreSQL');  
  
'postgresql':8 'кром':7 'нужн':2 'поддержива':5 'устанавлива':3  
  
(1 row)
```


to_tsquery

```
# SELECT to_tsquery('install | maintain');
```

```
'instal' | 'maintain'
```

```
(1 row)
```

```
# SELECT to_tsquery('russian', 'устанавливать & поддерживать');
```

```
'устанавлива' & 'поддержива'
```

```
(1 row)
```

plainto_tsquery & phraseto_tsquery

```
# SELECT plainto_tsquery('install maintain');
```

```
'instal' & 'maintain'
```

```
(1 row)
```

```
# SELECT phraseto_tsquery('russian', 'устанавливать поддерживать');
```

```
'устанавлива' <-> 'поддержива'
```

```
(1 row)
```

tsvector @@ tsquery

```
# SELECT to_tsvector('No need to install and maintain anything except PostgreSQL') @@ plainto_tsquery('install maintain') AS match;
```

match

t

Indexes: GIN or GiST?

GIN vs GiST:

- GIN
 - fast search, not very fast updates
 - better for static data
- GiST
 - slow search, faster updates
 - better for dynamic data

If you are not sure use GIN.

Practice: 1 / 3

CREATE TABLE IF NOT EXISTS

articles(id serial primary key, title varchar(128), content text);

-- https://meta.wikimedia.org/wiki/Data_dump_torrents#enwiki

-- <https://github.com/afiskon/postgresql-fts-example>

COPY articles FROM PROGRAM 'zcat /path/to/articles.copy.gz';

Practice: 2 / 3

```
CREATE OR REPLACE FUNCTION make_tsvector(title text, content text)
```

```
    RETURNS tsvector AS $$
```

```
BEGIN
```

```
    RETURN (setweight(to_tsvector('english', title), 'A') ||
```

```
        setweight(to_tsvector('english', content), 'B'));
```

```
END
```

```
$$ LANGUAGE 'plpgsql' IMMUTABLE;
```

Practice: 3 / 3

```
CREATE INDEX IF NOT EXISTS idx_fts_articles ON articles
```

```
  USING gin(make_tsvector(title, content));
```

```
SELECT id, title FROM articles WHERE
```

```
  make_tsvector(title, content) @@ to_tsquery('bjarne <-> stroustrup');
```

2470 | Binary search algorithm

2129 | Bell Labs

2130 | **Bjarne Stroustrup**

3665 | C (programming language)

ts_headline: 1 / 2

```
SELECT id, ts_headline(title, q) FROM articles,  
    to_tsquery('bjarne <-> stroustrup') AS q -- !!!  
WHERE make_tsvector(title, content) @@ q;
```

2470 | Binary search algorithm

2129 | Bell Labs

2130 | **Bjarne Stroustrup**

ts_headline: 2 / 2

```
SELECT id, ts_headline(title, q, 'StartSel=<em>, StopSel=</em>') -- !!!  
FROM articles, to_tsquery('bjarne <-> stroustrup') as q  
WHERE make_tsvector(title, content) @@ q;
```

2470 | Binary search algorithm

2129 | Bell Labs

2130 | **Bjarne Stroustrup**

ts_rank

```
SELECT id, ts_headline(title, q, 'StartSel=<em>, StopSel=</em>')  
FROM articles, to_tsquery('bjarne <-> stroustrup') as q  
WHERE make_tsvector(title, content) @@ q  
ORDER BY ts_rank(make_tsvector(title, content), q) DESC;
```

2130 | Bjarne Stroustrup

3665 | C (programming language)

6266 | Edsger W. Dijkstra

RUM

```
$ git clone git@github.com:postgrespro/rum.git
```

```
$ cd rum
```

```
$ USE_PGXS=1 make install
```

```
$ USE_PGXS=1 make installcheck
```

```
psql> CREATE EXTENSION rum;
```

Fuzzy Full Text Search

pg_trgm: 1 / 4

```
create extension pg_trgm;
```

```
create index articles_trgm_idx on articles using gin (title gin_trgm_ops);
```

pg_trgm: 2 / 4

```
select show_trgm(title) from articles limit 3;
```

```
show_trgm | {" a"," ac",acc,ble,cce,ces,com,eco,ess,ibl,ing,lec,mpu,...
```

```
show_trgm | {" a"," an",ana,arc,chi,his,ism,nar,rch,"sm "}
```

```
show_trgm | {" a"," af",afg,anh,ani,fg,gha,han,his,ist,nhi,nis,ory,...
```

pg_trgm: 3 / 4

```
select title, similarity(title, 'Straustrup') from articles where title % 'Straustrup';
```

```
-[ RECORD 1 ]-----
```

```
title          | Bjarne Stroustrup
```

```
similarity     | 0.35
```

pg_trgm: 4 / 4

```
psql> select show_limit();
```

```
-[ RECORD 1 ]---
```

```
show_limit | 0.3
```

```
psql> select set_limit(0.4);
```

```
-[ RECORD 1 ]--
```

```
set_limit | 0.4
```


pg_trgm: like / ilike queries

```
# explain select title from articles where title LIKE '%Stroustrup%';
```

QUERY PLAN

Bitmap Heap Scan on articles (cost=60.02..71.40 rows=3 width=16)

Recheck Cond: ((title)::text ~~ '%Stroustrup%')::text)

-> Bitmap Index Scan on articles_trgm_idx (cost=0.00..60.02 rows=3...

Index Cond: ((title)::text ~~ '%Stroustrup%')::text)

pg_trgm: regular expressions

```
# explain select title from articles where title ~* 'Stroustrup';
```

QUERY PLAN

Bitmap Heap Scan on articles (cost=60.02..71.40 rows=3 width=16)

Recheck Cond: ((title)::text ~* 'Stroustrup'::text)

-> Bitmap Index Scan on articles_trgm_idx (cost=0.00..60.02 rows=3...

Index Cond: ((title)::text ~* 'Stroustrup'::text)

See also

- The **pg_trgm** module provides functions and operators for determining the similarity of alphanumeric text based on trigram matching
 - <https://www.postgresql.org/docs/current/static/pgtrgm.html>
- Full Text Search support for JSON and JSONB
 - <https://www.depesz.com/2017/04/04/waiting-for-postgresql-10-full-text-search-support-for-json-and-jsonb/>
- RUM access method
 - <https://github.com/postgrespro/rum>

Thank you for your attention!

- <http://eax.me/>
- <http://devzen.ru/>

Bonus Slide!

GIN & arrays

```
create table vec_test(id serial primary key, tags int[]);
```

```
create index vec_test_gin on vec_test using gin(tags);
```

```
insert into vec_test (tags) values ('{111,222,333}');
```

```
select * from vec_test where '{111}' <@ tags;
```

```
select * from vec_test where '{111}' @> tags;
```

```
select * from vec_test where '{111}' = tags;
```

```
-- intersection is not empty
```

```
select * from vec_test where '{111}' && tags;
```