The Platinum phrase search with PostgreSQL

Andrii Soldatenko
17 June 2017
@a_soldatenko

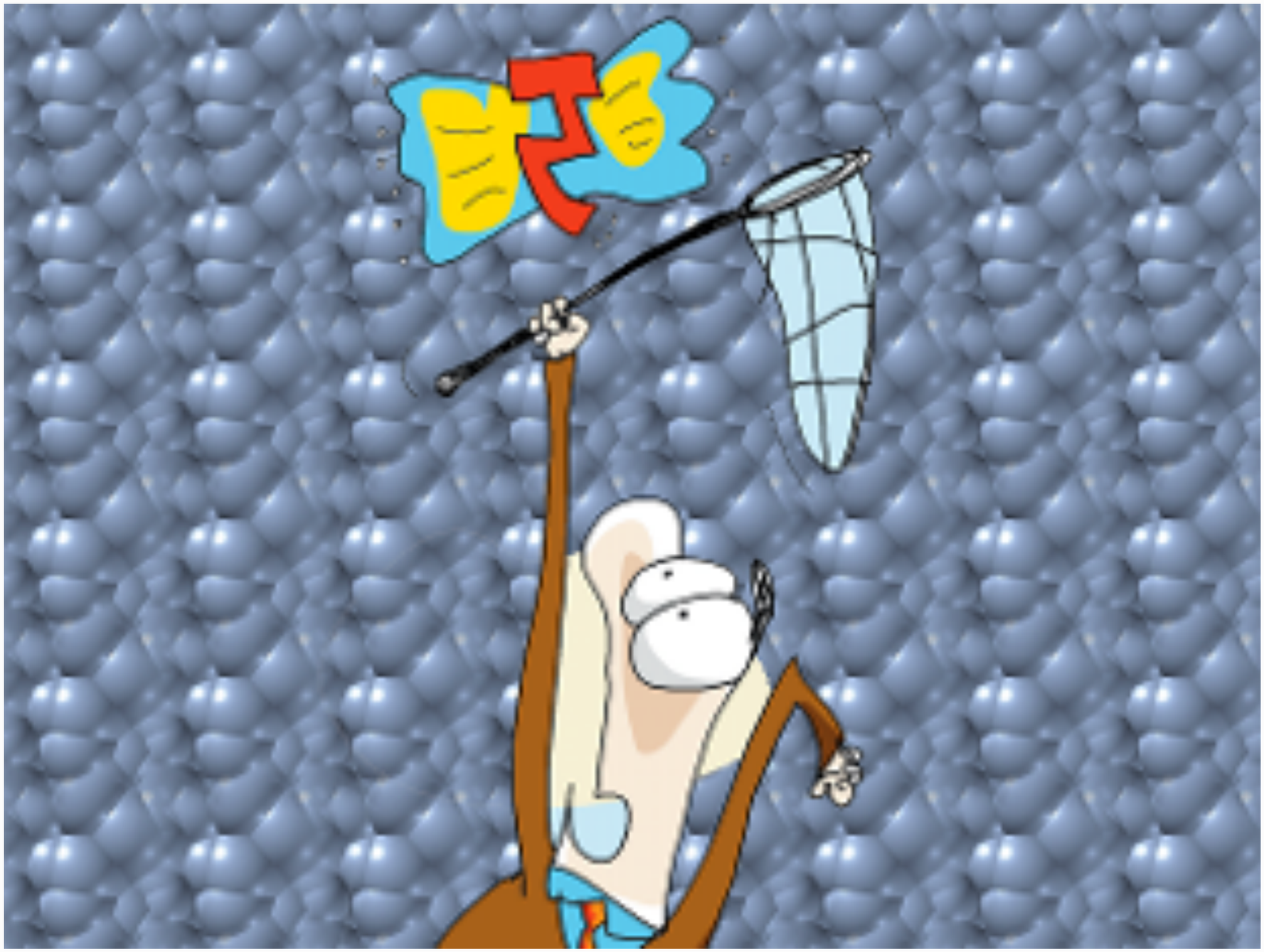
cat /etc/passwd

Andrii Soldatenko

- Senior Python Developer at Toptal
- Speaker at many PyCons and open source contributor
- blogger at https://asoldatenko.com

@a_soldatenko

Why Platinum?



@a_soldatenko

Text Search (grep)

➜ postgresql git:(master) time grep -r --exclude-dir=".git" "bigint" .
grep --color -r --exclude-dir=".git" "bigint" .
1.50s user 0.08s system 98% cpu 1.596 total

Processor 2.5 GHz Intel Core i7

Memory 16 GB 1600 MHz DDR3

Text Search (Platinum Searcher)

```
➜  postgresql git:(master) time pt bigint
0.28s user 0.15s system 448% cpu 0.094 total
```

Processor 2.5 GHz Intel Core i7

Memory 16 GB 1600 MHz DDR3

https://github.com/monochromegane/the_platinum_searcher

Text Search (ripgrep)

```
➜  postgresql git:(master) time rg -uu bigint
0.14s user 0.40s system 180% cpu 0.297 total
```

Processor 2.5 GHz Intel Core i7

Memory 16 GB 1600 MHz DDR3

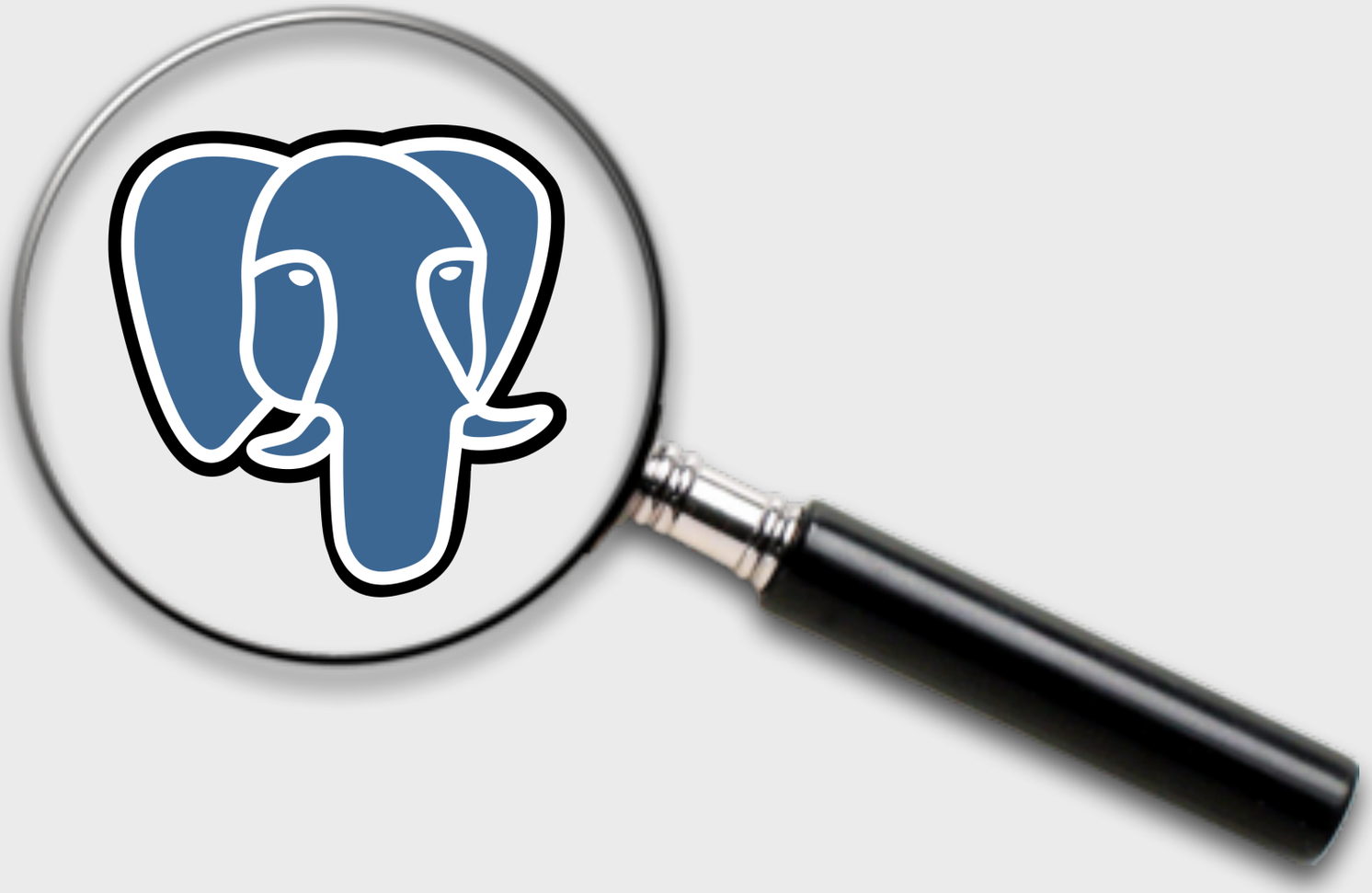
https://github.com/BurntSushi/ripgrep

Text Search benchmarks

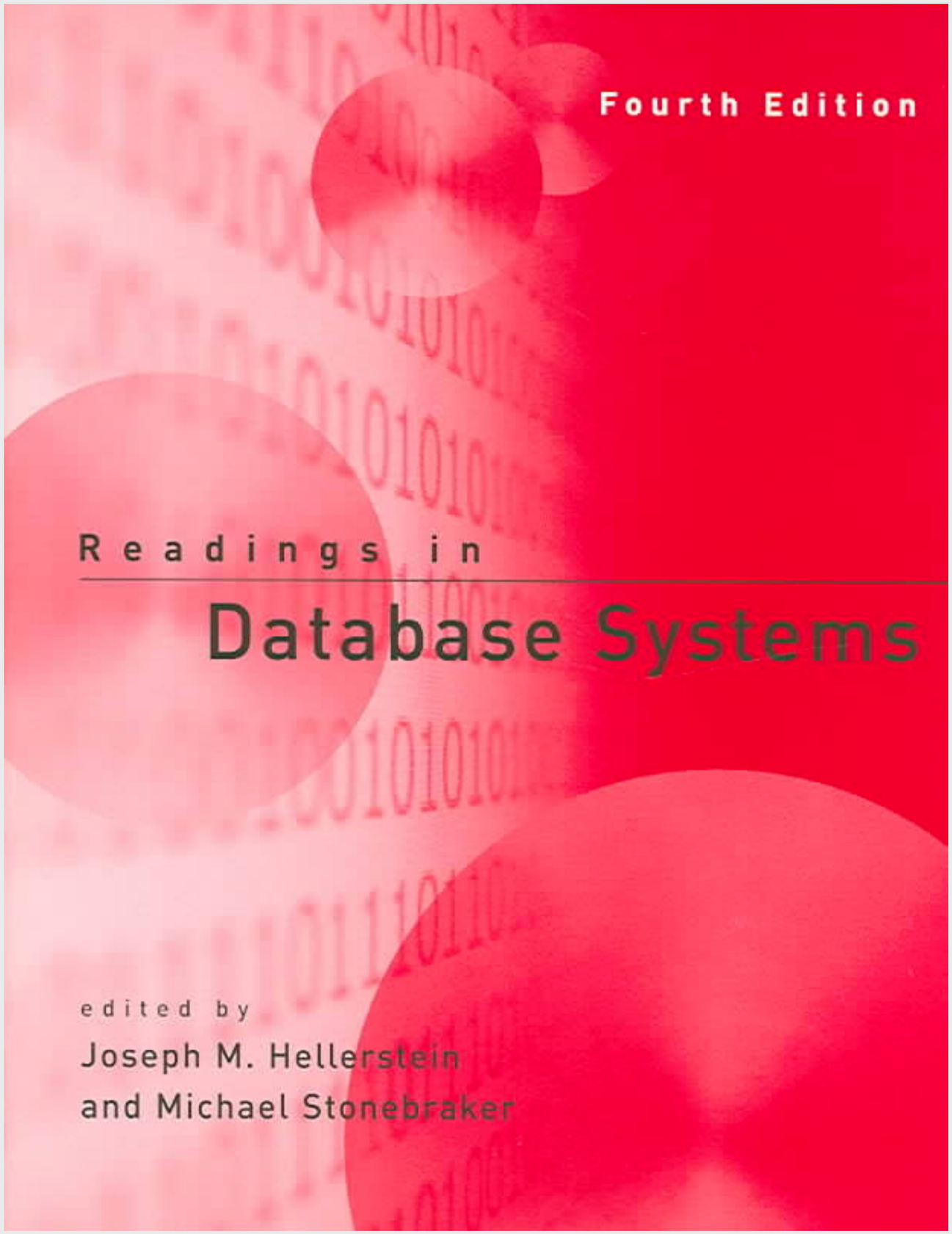
Tool	Command	Line count	Time
ripgrep (Unicode)	`rg -n -w '[A-Z]+_SUSPEND'`	450	**0.134s**
The Silver Searcher	`ag -w '[A-Z]+_SUSPEND'`	450	0.753s
git grep	`LC_ALL=C git grep -E -n -w '[A-Z]+_SUSPEND'`	450	0.823s
git grep (Unicode)	`LC_ALL=en_US.UTF-8 git grep -E -n -w '[A-Z]+_SUSPEND'`	450	2.880s
sift	`sift --git -n -w '[A-Z]+_SUSPEND'`	450	3.656s
The Platinum Searcher	`pt -w -e '[A-Z]+_SUSPEND'`	450	12.369s
ack	`ack -w '[A-Z]+_SUSPEND'`	1878	16.952s

http://blog.burntsushi.net/ripgrep/

PostgreSQL



Michael Stonebraker



http://www.redbook.io/

Full text search

Search index


BACK END DEVELOPER

PostgreSQL
Full Text Search

```
db=# SELECT to_tsvector('kiev devops
meetups') @@ to_tsquery('meetup');
 ?column?
----------
 t
(1 row)
```

‘kiev devops **meetups**’ <-> ‘**meetup**’

@a_soldatenko

Simple Full text search

```
SELECT to_tsvector('kiev devops meetup')
@@ to_tsquery('kiev meetup');
ERROR:  syntax error in tsquery: "kiev
meetup"
```

@a_soldatenko

Simple Full text search

```
SELECT to_tsvector('kiev devops
meetups')
@@ to_tsquery('meetup & kiev');
 ?column?
----------
 t
(1 row)
```

@a_soldatenko

iTunes Database

```
select count(*) from application;
  count
---------
 2366689
(1 row)
```

@a_soldatenko

More real example
Full text search

```
select title from application where search_vector @@
to_tsquery('Facebook') LIMIT 5;

                                  title
-------------------------------------------------------------------
 … for Facebook,WhatsApp,Twitter
 … Hair Sticker Editor for Facebook Anime Edition
 … Telegram, Kik, GroupMe, Viber, Snapchat, Facebook Me
 … T-Shirt for Facebook - …
 … Facebook - Check if someone … on Facebook
(5 rows)
```

@a_soldatenko

Origins of phrase search

text a <-> text b

@a_soldatenko

Phrase search

Teodor Sigaev and Oleg Bartunov

New FOLLOWED BY operator

```
select phraseto_tsquery(
'PostgreSQL allows searching for waxed
ducks.');
```

```
                phraseto_tsquery
-----------------------------------------------------
 'postgresql' <-> 'allow' <-> 'search' <2> 'wax' <-> 'duck'
(1 row)
```

@a_soldatenko

New FOLLOWED BY operator

```
SELECT
to_tsvector('kiev devops meetup') @@
to_tsquery('kiev <-> devops');
 ?column?
----------
 t
(1 row)
```

@a_soldatenko

New FOLLOWED BY operator

```
SELECT
to_tsvector('kiev devops meetup') @@
to_tsquery('kiev <-> meetup');
 ?column?
----------
 f
(1 row)
```

@a_soldatenko

New FOLLOWED BY operator

```
SELECT
to_tsvector('kiev devops meetup') @@
to_tsquery(
'kiev <-> devops & devops <-> meetup');
 ?column?
----------
 t
(1 row)
```

@a_soldatenko

New FOLLOWED BY operator

```
SELECT
to_tsvector('kiev devops meetup') @@
to_tsquery('kiev <2> meetup');
 ?column?
----------
 t
(1 row)
```

@a_soldatenko

near phrase search :)

```
select seller_name from application where
search_vector @@ plainto_tsquery('Facebook Inc')
LIMIT 5;
```

```
    seller_name
--------------------
 FunPokes, Inc.
 DATT JAPAN INC.
 Hoot Live, Inc
 Loytr Inc
 Facebook, Inc.
(5 rows)
```

unordered lexeme set

since PostgreSQL 9.6 we have ability to search for an exact phrase

@a_soldatenko

Phrase search

```
select seller_name from application
where search_vector @@
phraseto_tsquery('Facebook Inc')
LIMIT 5;
```

```
   seller_name
-----------------
 Facebook, Inc.
 Facebook, Inc.
 Facebook, Inc.
 Facebook, Inc.
 Facebook, Inc.
```

@a_soldatenko

As an user I want to see more relevant results

```
select title from application where search_vector @@
to_tsquery('Facebook') LIMIT 5;

                                  title
-------------------------------------------------------------------
 … for Facebook,WhatsApp,Twitter
 … Hair Sticker Editor for Facebook Anime Edition
 … Telegram, Kik, GroupMe, Viber, Snapchat, Facebook Me
 … T-Shirt for Facebook - …
 … Facebook - Check if someone … on Facebook
(5 rows)
```

@a_soldatenko

Ranking Search Results

```
# Ranks vectors based on the frequency
of their matching lexemes.
ts_rank(textsearch, query)

# This function computes the cover
density ranking for the given document
vector and query
ts_rank_cd(textsearch, query, 0 /* 1 2
4 8 16 32*/)
```

@a_soldatenko

Ranking Search Results

```
select left(title, 40),
ts_rank_cd(search_vector_title,
phraseto_tsquery('instagram')) as rank
from application
 where search_vector_title @@
phraseto_tsquery('instagram')
order by rank
LIMIT 5;
```

left	rank
Love Frames : Share your valentine photo	0.1
FullSized Insta - Square Ready Fotos for	0.1
InstaClean for Instagram -Cleaner Mass D	0.1
Stickers Free for WhatsApp, Telegram, Ki	0.1
Pip Camera - Photo Collage Maker For Ins	0.1

divides the rank by the document length

```
select left(title, 40),
ts_rank_cd(search_vector_title,
phraseto_tsquery('facebook'), 2) as rank from
application where search_vector_title @@
phraseto_tsquery('facebook') order by rank desc
LIMIT 5;
```

left	rank
Facebook	0.1
Who Deleted Me? for Facebook	0.05
Location for Facebook	0.05
MessengerApp for Facebook	0.05
Picturito for Facebook	0.05

(5 rows)

@a_soldatenko

Performance

@a_soldatenko

GIN

@a_soldatenko

Generalized Inverted index

Term	Doc_1	Doc_2
Quick		X
The	X	
brown	X	X
dog	X	
dogs		X
fox	X	
foxes		X
in		X
jumped	X	
lazy	X	X
leap		X
over	X	X
quick	X	
summer		X
the	X	

@a_soldatenko

GIN

```
CREATE INDEX name ON table USING GIN
(column);
```

@a_soldatenko

GIN

@a_soldatenko

GIN

- Slow ranking.

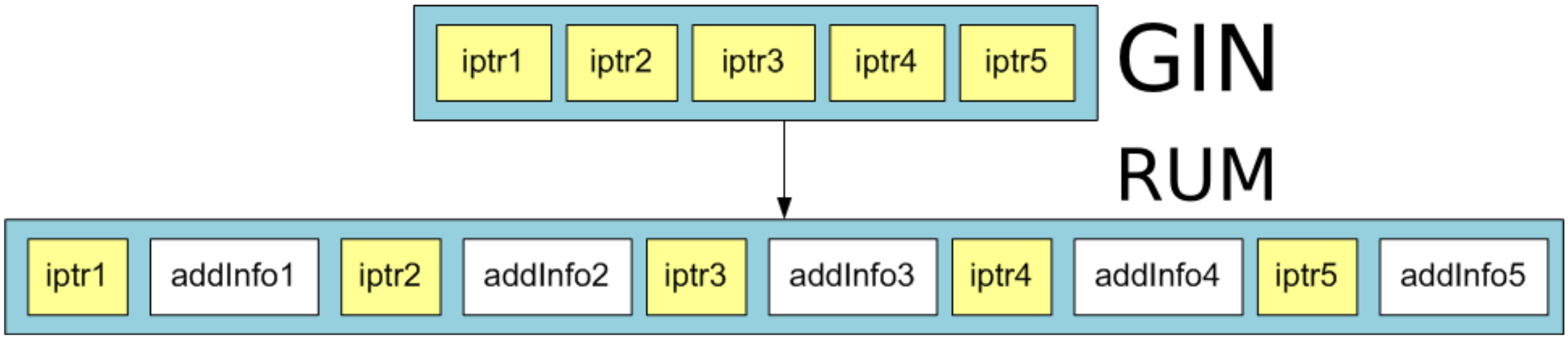
- Slow phrase search

- Slow ordering by timestamp

RUM

@a_soldatenko

RUM index



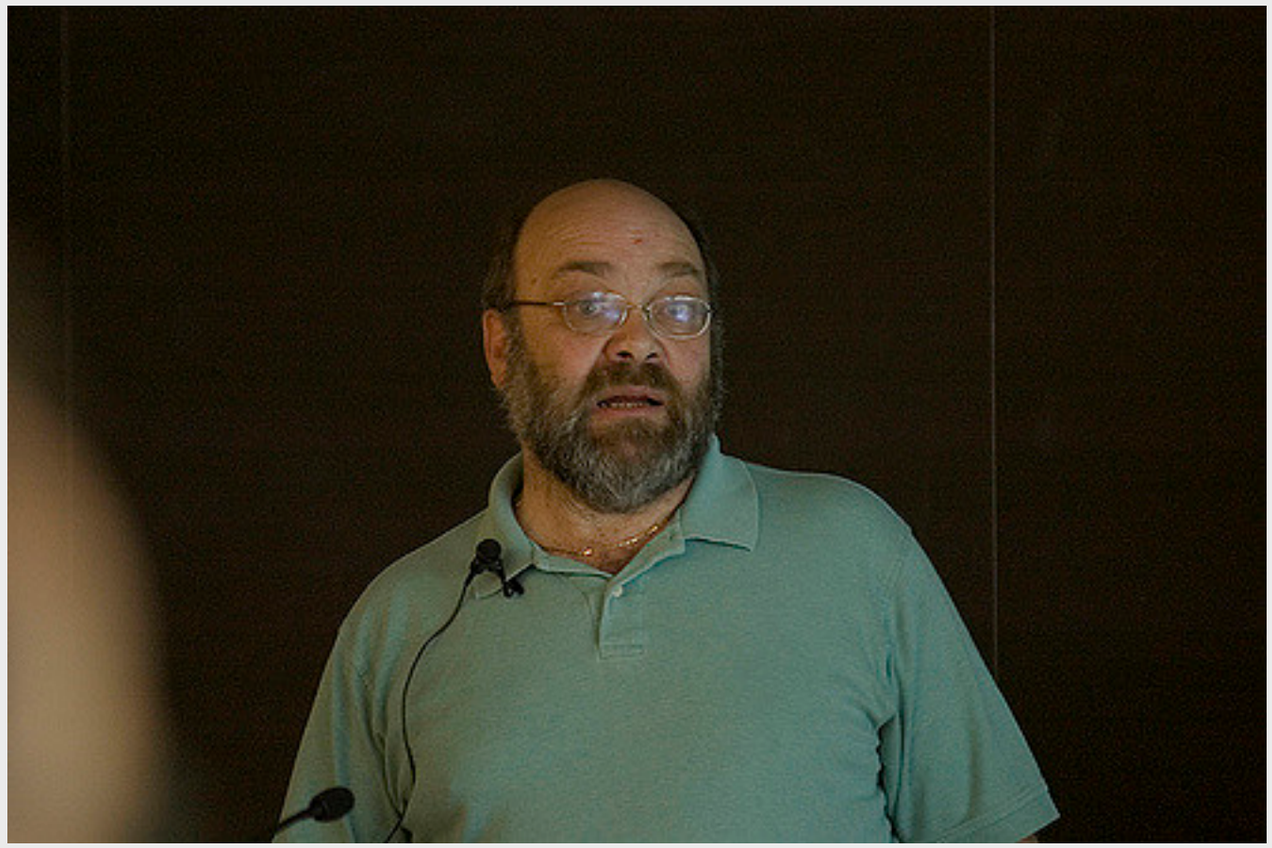
https://github.com/postgrespro/rum

STEVEN SPIELBERG PRESENTS

BACK TO THE FUTURE™

A ROBERT ZEMECKIS FILM

Andrew Dunstan committed patch:



http://git.postgresql.org/pg/commitdiff/e306df7f9cd6b4433273e006df11bdc966b7079e

PostgreSQL 10 Full text search support for json and jsonb

Release date: August 10th, 2017

```
$ select id, jsonb_pretty(payload) from test;
 id |                                                jsonb_pretty
----+-------------------------------------------------------------------------------------------------------
  1 | {                                                                                                    +
    |     "glossary": {                                                                                    +
    |         "title": "example glossary",                                                                 +
    |         "GlossDiv": {                                                                                +
    |             "title": "S",                                                                            +
    |             "GlossList": {                                                                           +
    |                 "GlossEntry": {                                                                      +
    |                     "ID": "SGML",                                                                    +
    |                     "Abbrev": "ISO 8879:1986",                                                       +
    |                     "SortAs": "SGML",                                                                +
    |                     "Acronym": "SGML",                                                               +
    |                     "GlossDef": {                                                                    +
    |                         "para": "A meta-markup language, used to create markup languages such as DocBook.",+
    |                         "GlossSeeAlso": [                                                            +
    |                             "GML",                                                                   +
    |                             "XML"                                                                    +
    |                         ]                                                                            +
    |                     },                                                                               +
    |                     "GlossSee": "markup",                                                            +
    |                     "GlossTerm": "Standard Generalized Markup Language"                              +
    |                 }                                                                                    +
    |             }                                                                                        +
    |         }                                                                                            +
    |     }                                                                                                +
    | }
(1 row)
```

Full text search on json

```
select to_tsvector('english', payload) from test;
                                                to_tsvector
-----------------------------------------------------------------------------------------------------
 '1986':8 '8879':7 'creat':21 'docbook':26
'exampl':1 'general':35 'glossari':2.
. 'gml':28 'iso':6 'languag':18,23,37 'markup':
17,22,32,36 'meta':16 'meta-mark.
.up':15 'sgml':4,10,12 'standard':34 'use':19
'xml':30
(1 row)
```


django

Django 2.0

#28194	Add search rank cd function and normalization for Postgres full text search	assigned	**Andrii Soldatenko**	New feature	contrib.postgres	1.11
#28077	Allow specifying an operator class for GinIndex()	new		New feature	contrib.postgres	1.11
#28041	Postgres prefix searching for full text search	assigned	Joe Tsoi	New feature	contrib.postgres	1.10
#27899	Phrase search query for Postgres full text search	assigned	**Andrii Soldatenko**	New feature	contrib.postgres	master

Django hardcoded ts functions

```
class SearchQuery(SearchQueryCombinable, Value):
    def as_sql(self, compiler, connection):
        ...
            template =
'plainto_tsquery({}::regconfig,
%s)'.format(config_sql)
```

Django ORM cd_ranks

```
class SearchRankCD(SearchRank):
    function = 'ts_rank_cd'

    def __init__(self, vector, query, normalization=0, **extra):
        super(SearchRank, self).__init__(
            vector, query, normalization, **extra)

query = SearchQuery('messenger')

Application.objects.annotate(
    rank=SearchRankCD(
        F('search_vector_title'), query,
        normalization=2)  # 2 divides the rank by the document length
).filter(search_vector_title=query).order_by('-rank')
```

Conclusions

- PostgreSQL FTS combine with relation queries.
- phrase search works fast (ms)
- don’t forget to contribute if you fix something

Thank You

https://asoldatenko.com

@a_soldatenko

Questions

toptal

@a_soldatenko

We are hiring

https://www.toptal.com/#connect-fantastic-computer-engineers