

How elephants survive in big data environments?

Andrii Soldatenko

28 October 2017

 @a_soldatenko

cat /etc/passwd

Andrii
Soldatenko



- Senior Python/Go Developer at Toptal
- Speaker and open source contributor
- blogger at <https://asoldatenko.com>

Idea



Too many information



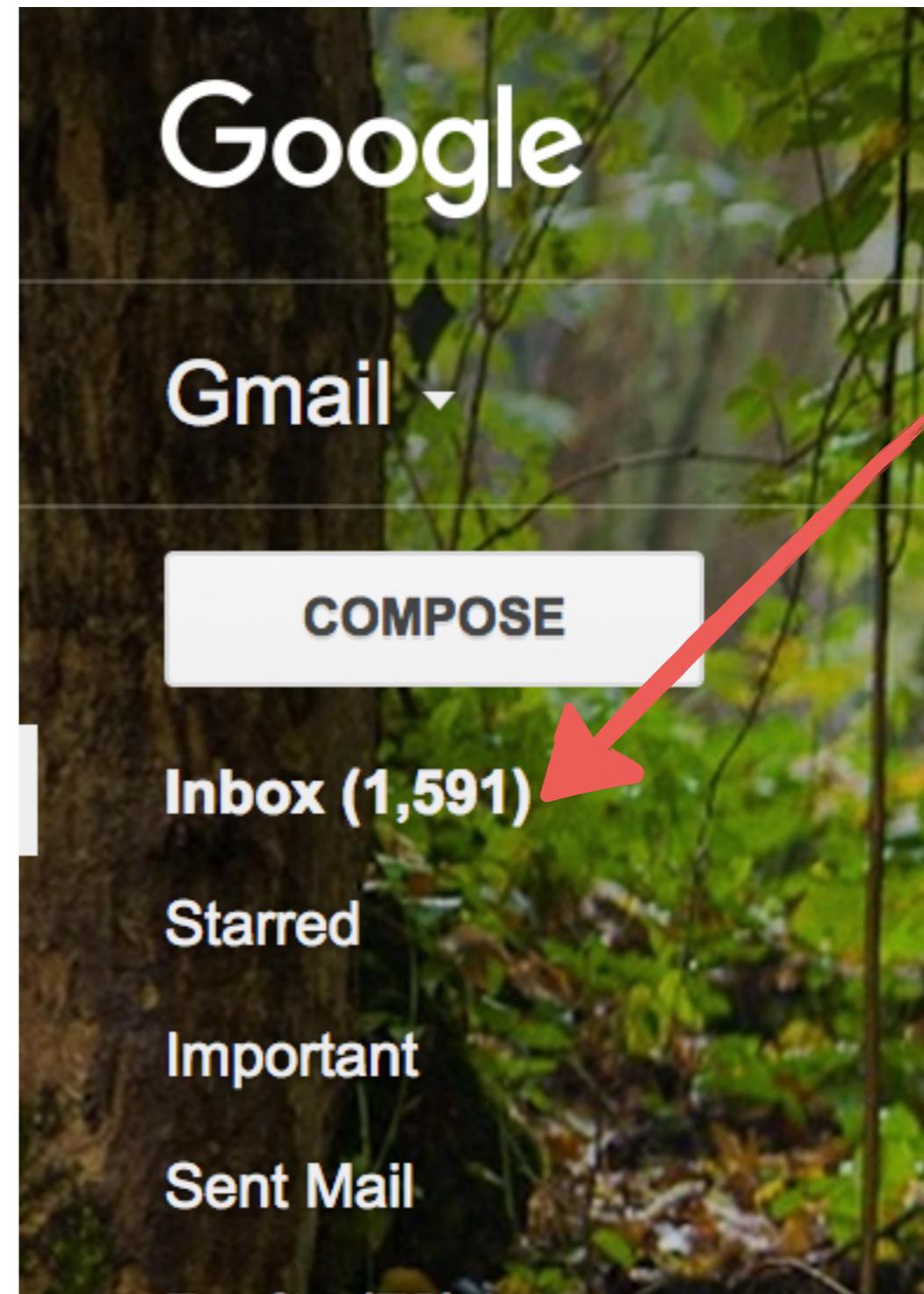
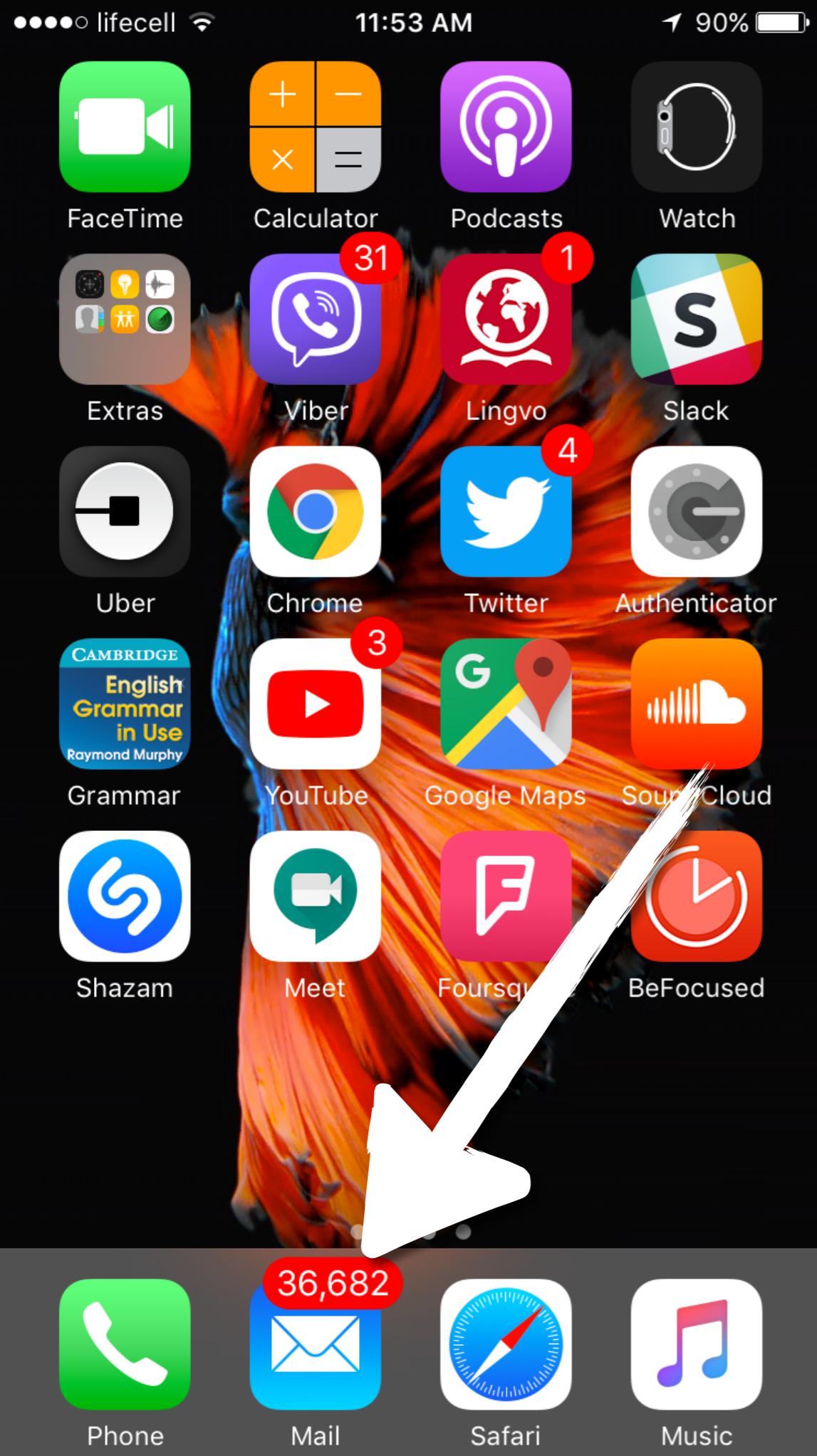
Information Explosion



Text Search

```
$ grep --recursive foo books/
```

<https://www.gnu.org/software/grep/manual/grep.html>



email lists
github notifications
jira notifications





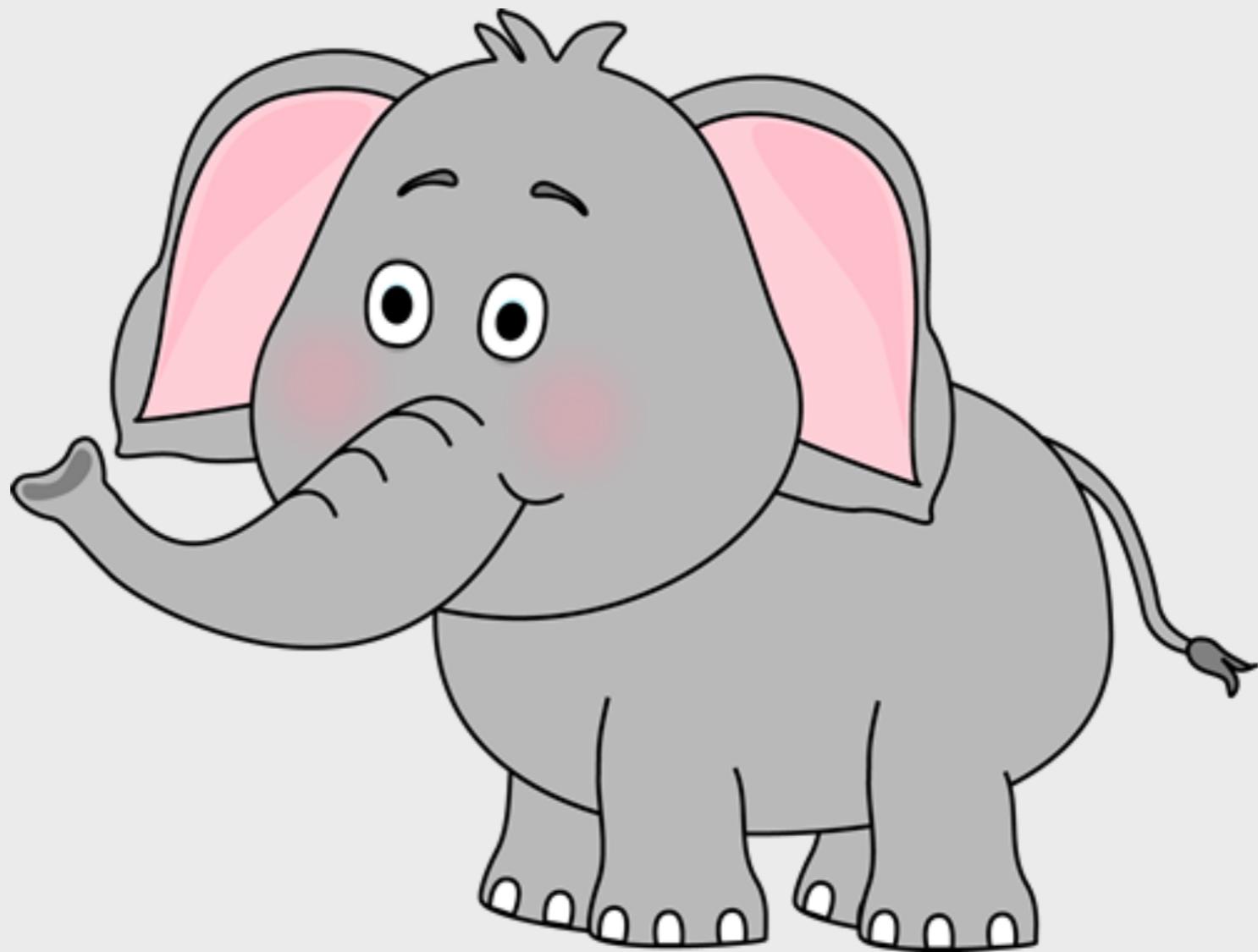
WE NEED TO GO

DEEPER

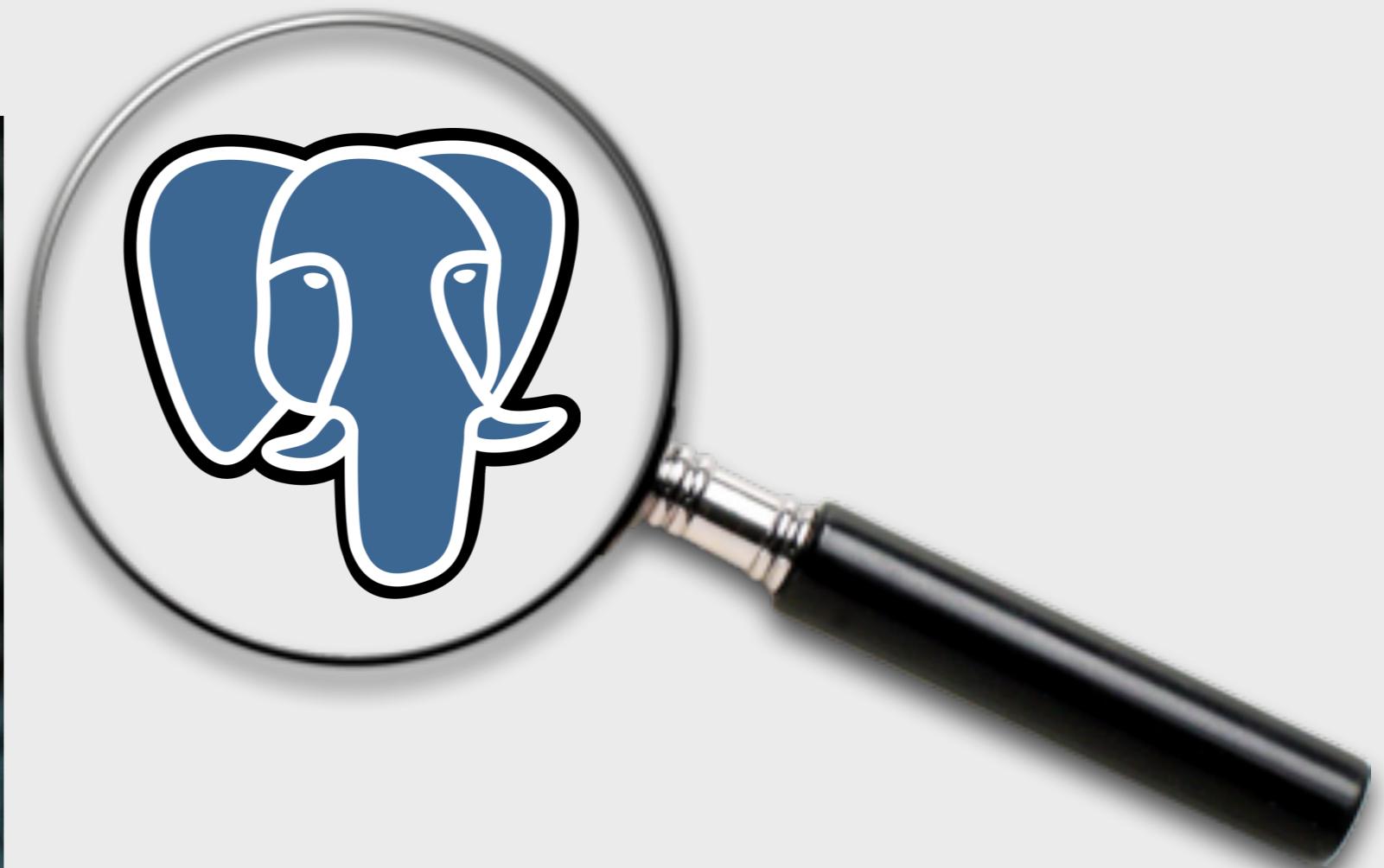


Text Search benchmarks

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



PostgreSQL



Michael Stonebraker

Fourth Edition

Readings in

Database Systems

edited by

Joseph M. Hellerstein

and Michael Stonebraker

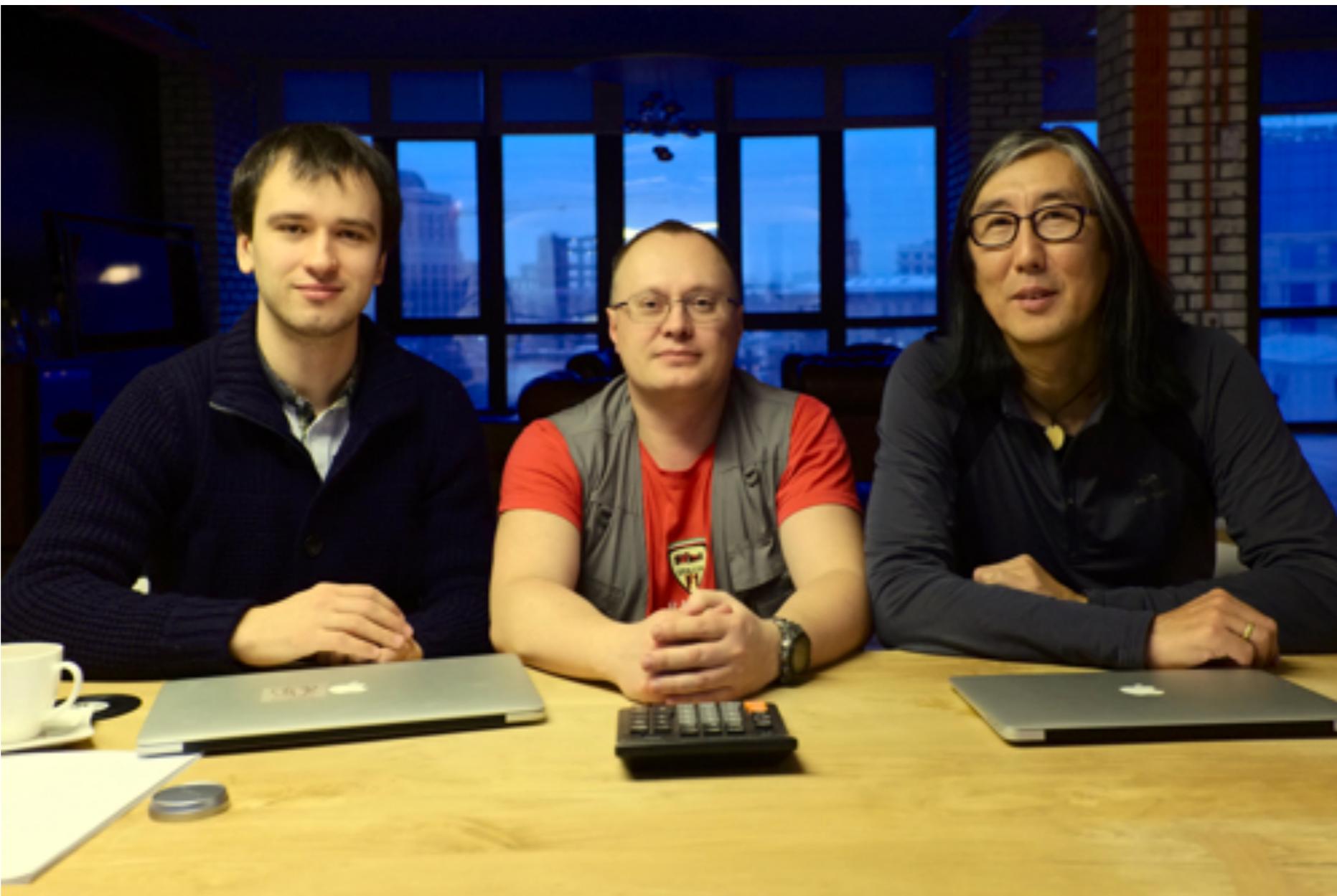


<http://www.redbook.io/>



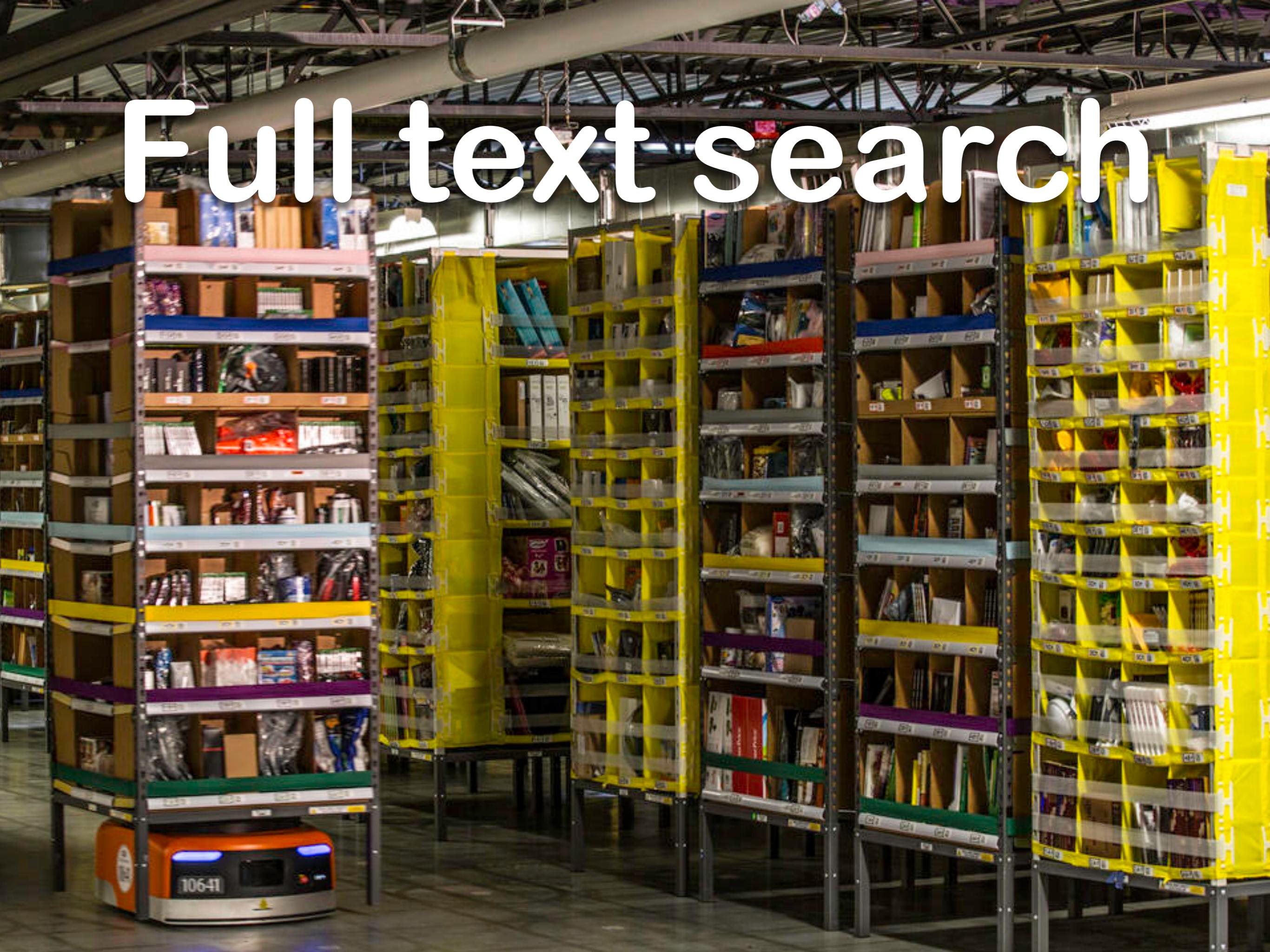
open.az





Александр Коротков, Федор Сигаев и Олег Бартунов
(слева направо)

Full text search



Search index

Symbols

32gb Heap boundary, 642

A

ACID transactions, 545, 556

action, in bulk requests, 57, 69

ad hoc searches, 15

aggregations, 20, 417

 aggs parameter, 424

 and analysis, 483

 approximate, 457

 cardinality, 458

 percentiles, 462

basic example

 adding a metric, 426

 adding extra metrics, 429

 buckets nested in other buckets, 427

buckets, 419

 combining buckets and metrics, 420

 metrics, 420

limiting memory usage, 487

 fielddata circuit breaker, 490

 fielddata size, 488

 moitoring fielddata, 489

managing efficient memory usage, 507

nested, 567

 reverse_nested aggregation, 568

operating alongside search requests, 418

preventing combinatorial explosions, 500

 depth-first versus breadth-first, 502

returning empty buckets, 439

scoping, 445

 global bucket, 447

Significant Terms, 471

significant terms

iTunes Database

```
select count(*) from application;  
count
```

```
2,627,949  
( 1 row )
```

PostgreSQL

Full Text Search

Demo

Simple Full text search

```
SELECT to_tsvector('awesome  
facebook app')  
@@ to_tsquery('facebook');  
?column?
```

```
t  
( 1 row)
```

More real example

Full text search

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

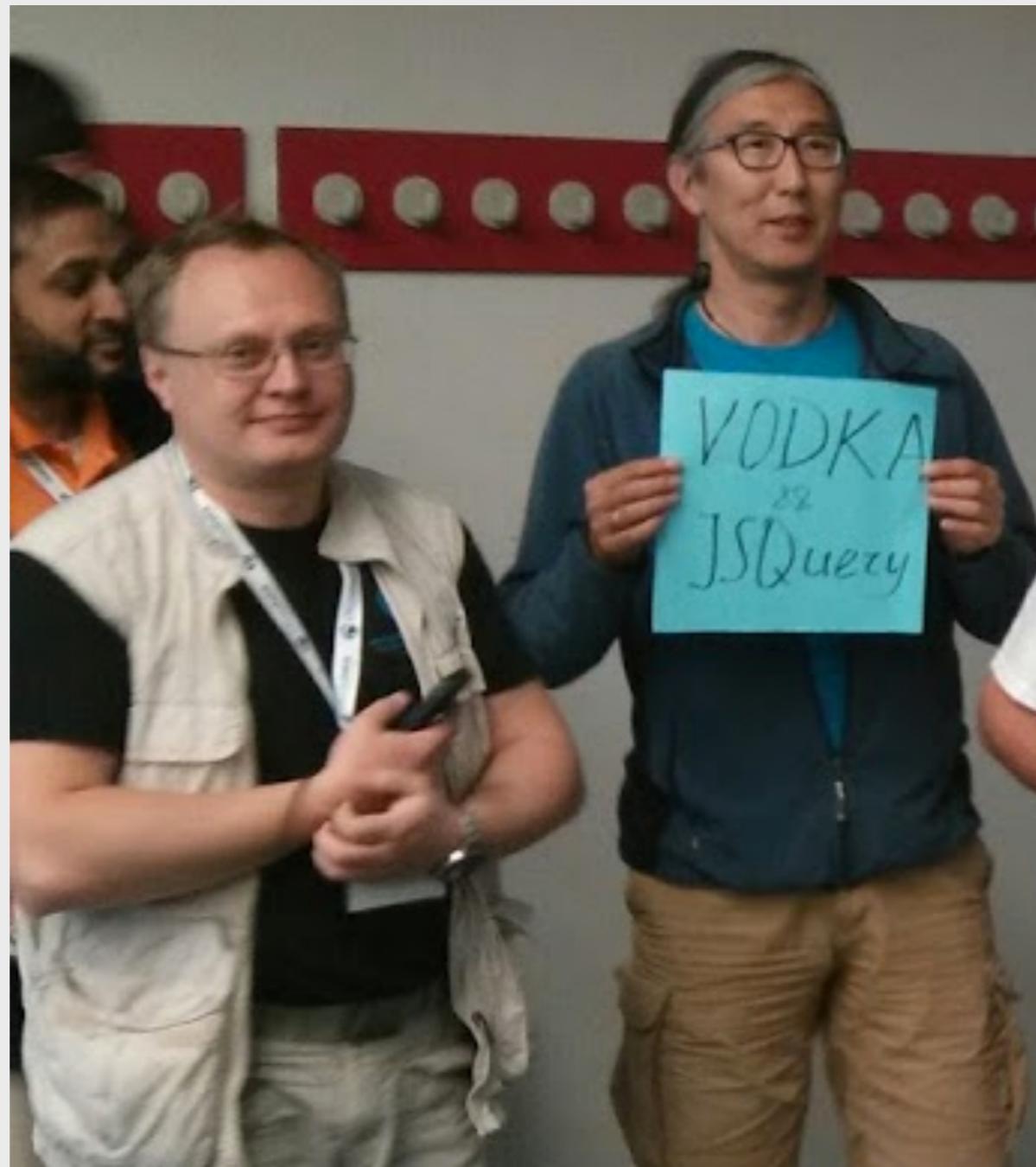
title

```
... 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)
```

Origins of phrase search

text a <-> text b

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 )
```

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

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.

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)
```

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 */)
```

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)	

Performance

GIN



Generalized Inverted index

1	Term	Doc_1	Doc_2
2		-----	
3	Quick		X
4	The	X	
5	brown	X	X
6	dog	X	
7	dogs		X
8	fox	X	
9	foxes		X
10	in		X
11	jumped	X	
12	lazy	X	X
13	leap		X
14	over	X	X
15	quick	X	
16	summer		X
17	the	X	
18		-----	



GIN

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

GIN



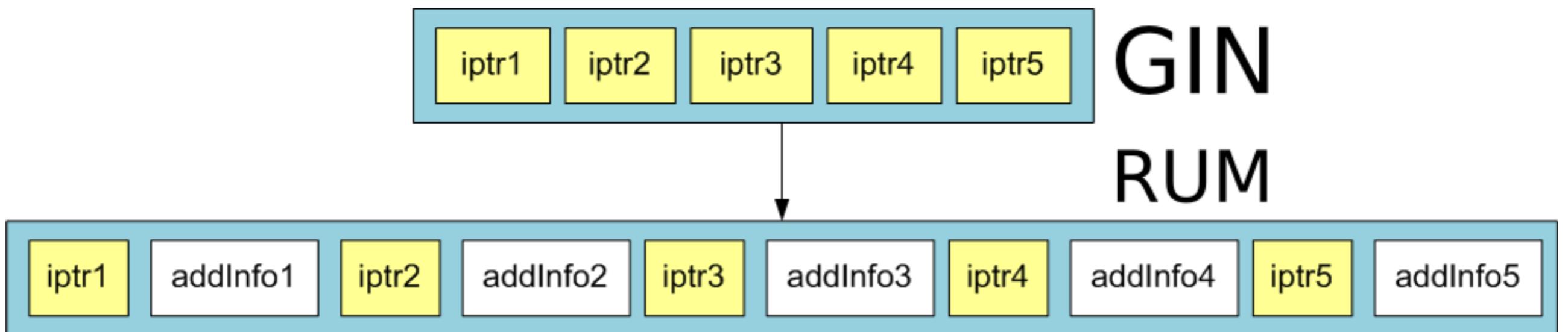
GIN

- Slow ranking.
- Slow phrase search
- Slow ordering by timestamp

RUM



RUM index



<https://github.com/postgrespro/rum>

Limit	Value
Maximum Database Size	Unlimited
Maximum Table Size	32 TB
Maximum Row Size	1.6 TB
Maximum Field Size	1 GB
Maximum Rows per Table	Unlimited
Maximum Columns per Table	250 - 1600
Maximum Indexes per Table	Unlimited

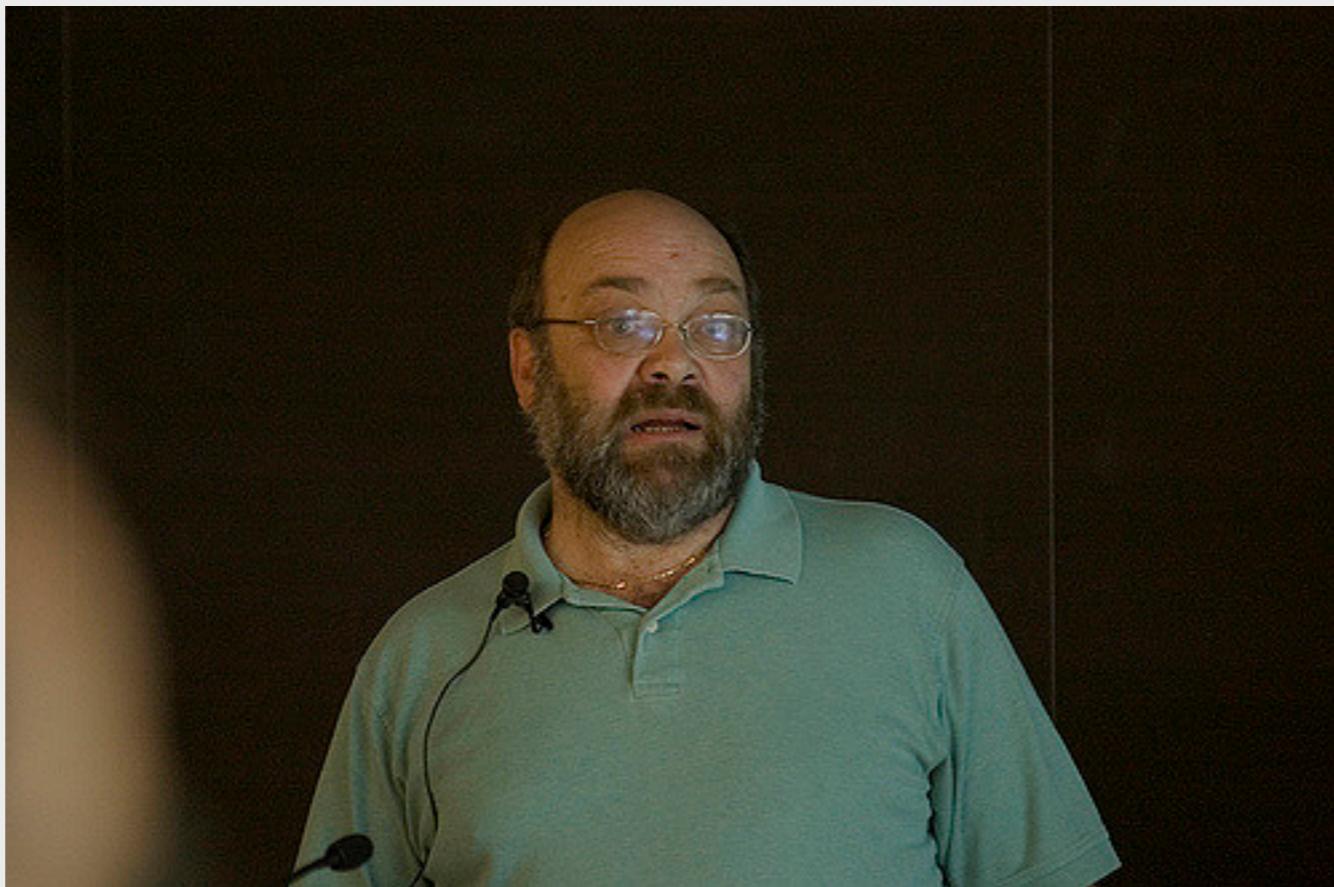
STEVEN SPIELBERG PRESENTS

BACK TO THE FUTURE

A ROBERT ZEMECKIS FILM

The logo features the title 'BACK TO THE FUTURE' in a bold, stylized font. The letters are primarily orange and yellow, with black outlines. The word 'BACK' is positioned above 'TO THE' and 'FUTURE'. A large, dynamic arrow graphic originates from the letter 'K' in 'BACK' and points towards the letter 'T' in 'TO', creating a sense of motion. The entire title is set against a black background. Below the main title, the subtitle 'A ROBERT ZEMECKIS FILM' is written in a smaller, italicized, and slightly less bold font.

Andrew Dunstan committed patch:



[http://git.postgresql.org/pg/commitdiff/
e306df7f9cd6b4433273e006df11bdc966b7079e](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

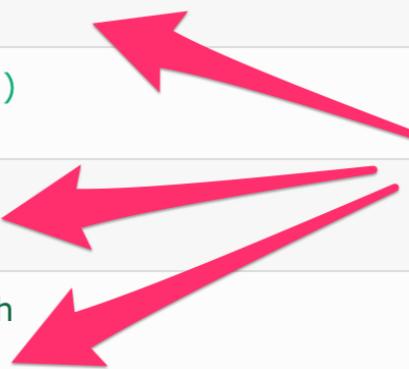
```
select to_tsvector('english', payload) from test;
          to_tsvector
-----
'1986':8 '8879':7 'creat':21 'docbook':26
'exAMPL':1 'general':35 'glossari':2.
. 'gml':28 'iso':6 'languag':18,23,37 'markup':
17,22,32,36 'meta':16 'meta-mark.
.up':15 'sgml':4,10,12 'standard':34 'use':19
'xml':30
(1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

