Последние новости полнотекстового поиска

Федор Сигаев

www.postgrespro.ru

Agenda

- Full text search in PostgreSQL
- Проблемы
- Новое:
  - CREATE INDEX … USING RUM
  - Phrase search
  - Inverse FTS
  - Dictionaries as extensions
  - Dictionaries in shared memory
  - Tsvector editing functions

FTS in Databases

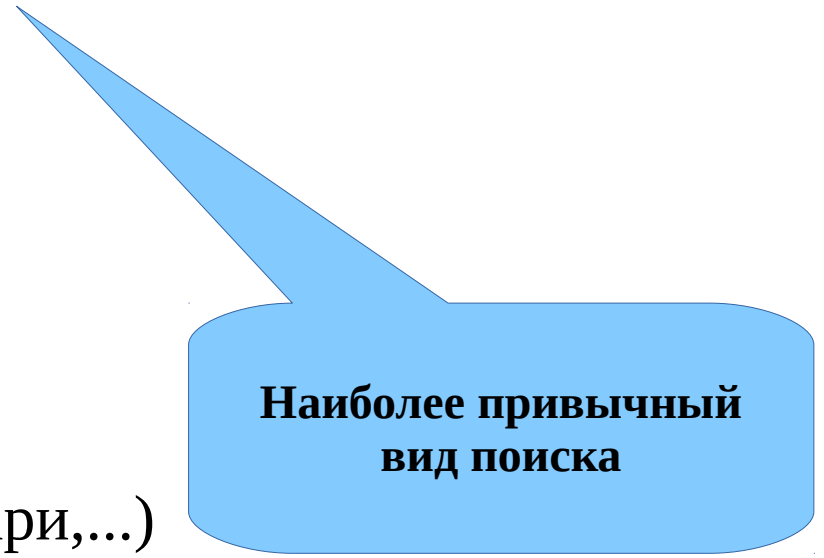
Полнотекстовый поиск

- найти документы удовлетворяющие запросу
- отсортировать их в некотором порядке

Найти документы содержащие **все** слова из запроса и вернуть их отсортированными по похожести

Требования к FTS

- **полная интеграция с СУБД**
  - транзакционность
  - конкурентный доступ
  - восстановление после сбоев
  - online индекс
- Конфигурируемость (парсеры, словари,...)
- Масштабируемость



Test Search Operators

“Традиционные” операции текстового поиска
(TEXT op TEXT, op - ~, ~*, LIKE, ILIKE)

```
=# select title  from apod where title ~* 'x-ray' limit 5;
                 title
-----------------------------------------
 The X-Ray Moon
 Vela Supernova Remnant in X-ray
 Tycho's Supernova Remnant in X-ray
 ASCA X-Ray Observatory
 Unexpected X-rays from Comet Hyakutake
(5 rows)

=# select title  from apod where title ilike '%x-ray%' limit 5;
```

What's wrong?

Нет поддержки лингвистики

- что есть слово ?
- что индексировать ?
- «нормализация» слов
- стоп-слова (noise-words)

Нет релевантности

- все документы одинаково «похожи»

Медленно, документы каждый раз сканируются

В 9.3+ появилась индексная поддержка (pg_trgm)

```
select * from man_lines where man_line ~* '(?:(?:p(?:ostgres(?:ql)?|g?sql)|sql)) (?:(?:(?:mak|us)e|do|is))';
```

FTS in PostgreSQL

- OpenFTS — 2000, Pg as a storage
- GiST index — 2000, thanks Rambler
- Tsearch — 2001, contrib:no ranking
- Tsearch2 — 2003, contrib:config
- GIN —2006, thanks, JFG Networks
- FTS — 2006, in-core, thanks,EnterpriseDB
- Now — Postgres Professional

FTS in PostgreSQL

tsvector - хранилище для документов, оптимизированное для поиска

- отсортированный массив лексем
- позиционная информация
- структурная информация (важность)

tsquery - текстовый тип для запроса с логическими операторами & | ! ()

Полнотекстовый оператор:
tsvector @@ tsquery

Операторы @>, <@ для tsquery

Функции: to_tsvector, to_tsquery, plainto_tsquery, ts_lexize, ts_debug, ts_stat, ts_rewrite,ts_headline, ts_rank, ts_rank_cd, setweight

Индексы: GiST, GIN

FTS summary

- Полнотесктовый поиск в PostgreSQL — это гибкий движок, но он больше, чем законченное решение
- Это набор набор «кирпичиков»
  - Парсер
  - Словари
  - + вся мощь SQL
  - Tsvector — как специализированный storage

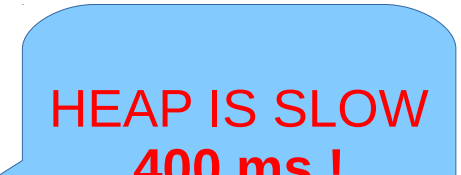
Химические формулы или геном

Some FTS problems #1

156676 Wikipedia articles:

- Search is fast, ranking is slow.

```
postgres=# explain analyze
SELECT docid, ts_rank(text_vector, to_tsquery('english', 'title')) AS rank
FROM ti2
WHERE text_vector @@ to_tsquery('english', 'title')
ORDER BY rank DESC
LIMIT 3;
```




```
 Limit  (cost=8087.40..8087.41 rows=3 width=282) (actual time=433.749..433.752 rows=3 loops=
   ->  Sort  (cost=8087.40..8206.63 rows=47692 width=282)
(actual time=433.749..433.749 rows=3 loops=1)
         Sort Key: (ts_rank(text_vector, '''titl'''::tsquery))
         Sort Method: top-N heapsort  Memory: 25kB
         ->  Bitmap Heap Scan on ti2  (cost=529.61..7470.99 rows=47692 width=282)
(actual time=15.094..423.452 rows=47855 loops=1)
               Recheck Cond: (text_vector @@ '''titl'''::tsquery)
               ->  Bitmap Index Scan on ti2_index  (cost=0.00..517.69 rows=47692 width=0)
(actual time=13.736..13.736 rows=47855 loops=1)
                     Index Cond: (text_vector @@ '''titl'''::tsquery)
 Total runtime: 433.787 ms
```

Some FTS problems #2

- Нет фразового поиска
  - «A & B» то же самое что и «B & A»
  - Комбинация полнотекстового поиска и regexp может помочь в простых случаях. Но не быстро.

Some FTS problems #3

- Найти документы, близкие или ограниченные по дате.

```
select sent,  subject from pglist where fts @@ to_tsquery('english', 'tom
& lane') order abs(sent - '2000-01-01'::timestamp) asc limit 5;

QUERY PLAN
------------------------------------------------------------------------------------------------------------------------
 Limit (actual time=545.560..545.560 rows=5 loops=1)
   ->  Sort (actual time=545.559..545.559 rows=5 loops=1)
         Sort Key: (CASE WHEN ((sent - '2000-01-01 00:00:00'::timestamp without time zone) < '00:00:00'::interval) THEN (-
(sent - '2000-01-01 00:00:00'::timestamp without time zone)) ELSE (sent - '2000-01-01 00:00:00'::timestamp without time zone)
END)
         Sort Method: top-N heapsort  Memory: 25kB
         ->  Bitmap Heap Scan on pglist (actual time=87.545..507.897 rows=222813 loops=1)
               Recheck Cond: (fts @@ '''tom'' & ''lane'''::tsquery)
               Heap Blocks: exact=105992
               ->  Bitmap Index Scan on pglist_gin_idx (actual time=57.932..57.932 rows=222813 loops=1)
                     Index Cond: (fts @@ '''tom'' & ''lane'''::tsquery)
 Planning time: 0.376 ms
 Execution time: 545.744 ms

(11 rows)
         sent          |                                 subject
-----------------------+-----------------------------------------------------------
 1999-12-31 13:52:55 | Re: [HACKERS] LIKE fixed(?) for non-ASCII collation orders
 2000-01-01 11:33:10 | Re: [HACKERS] dubious improvement in new psql
 1999-12-31 10:42:53 | Re: [HACKERS] LIKE fixed(?) for non-ASCII collation orders
 2000-01-01 13:49:11 | Re: [HACKERS] dubious improvement in new psql
 1999-12-31 09:58:53 | Re: [HACKERS] LIKE fixed(?) for non-ASCII collation orders
(5 rows)

Time: 568.357 ms
```

Some FTS problems #4

- Словари загружаются медленно
- Сконфигурить даже уже существующий словарь может быть непросто.

```
time for i in {1..10}; do echo $i; psql postgres -c "select
ts_lexize('english_hunspell', 'evening')" > /dev/null;  done
1
2
3
4
5
6
7
8
9
10

real    0m0.656s
user    0m0.015s
sys 0m0.031s
```

For russian hunspell dictionary:

real 0m3.809s
user0m0.015s
sys 0m0.029s

Each session «eats» 20MB !

PROFESSIONAL Postgres

There are always more other problems !

Улучшим GIN

- Добавим нужную информацию в postiong tree (позиции лексем или timestamp)
- И используем ее для сортировки (да и для ограничения поиска)

PROFESSIONAL Postgres

No positions in index !

Inverted Index in PostgreSQL

ENTRY TREE

Report Index

A

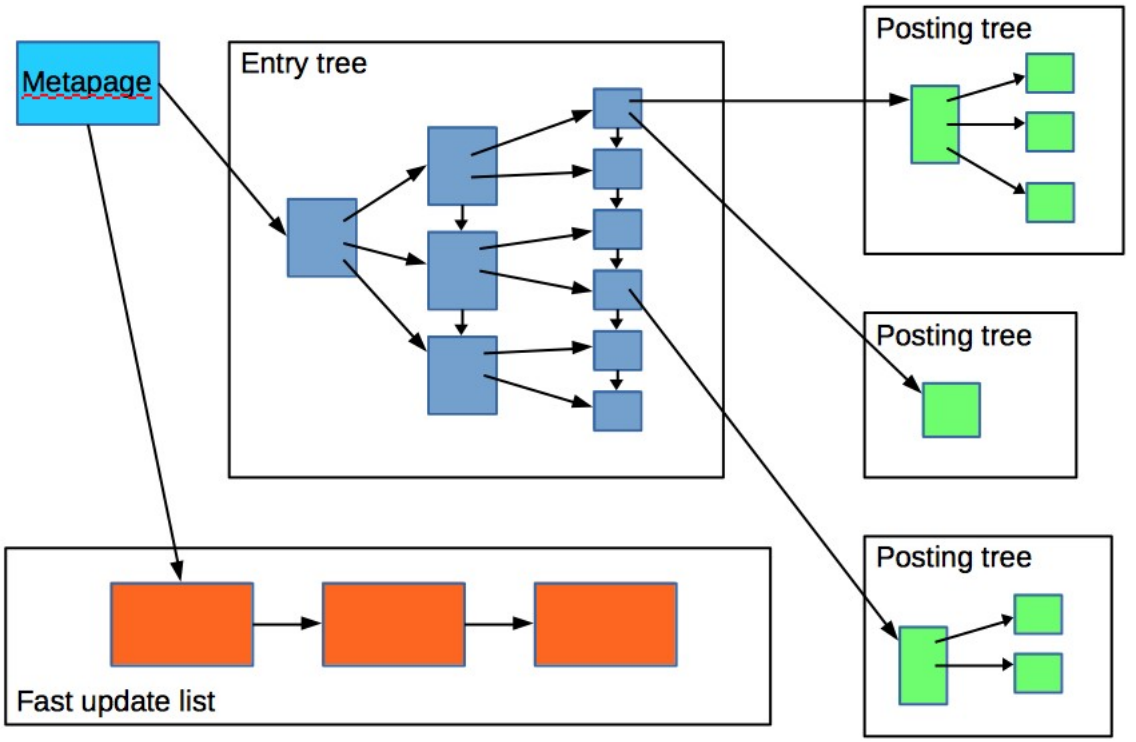
abrasives, 27
acceleration measurement, 58
accelerometers, 5, 10, 25, 28, 30, 36, 58, 59, 61, 73, 74
actuators, 4, 37, 46, 49
adaptive Kalman filters, 60, 61
adhesion, 63, 64
adhesive bonding, 15
adsorption, 44
aerodynamics, 29
aerospace instrumentation, 61
aerospace propulsion, 52
aerospace robotics, 68
aluminium, 17
amorphous state, 67
angular velocity measurement, 58
antenna phased arrays, 41, 46, 66
argon, 21
assembling, 22
atomic force microscopy, 13, 27, 35
atomic layer deposition, 15
attitude control, 60, 61
attitude measurement, 59, 61
automatic test equipment, 71
automatic testing, 24

Posting list
Posting tree

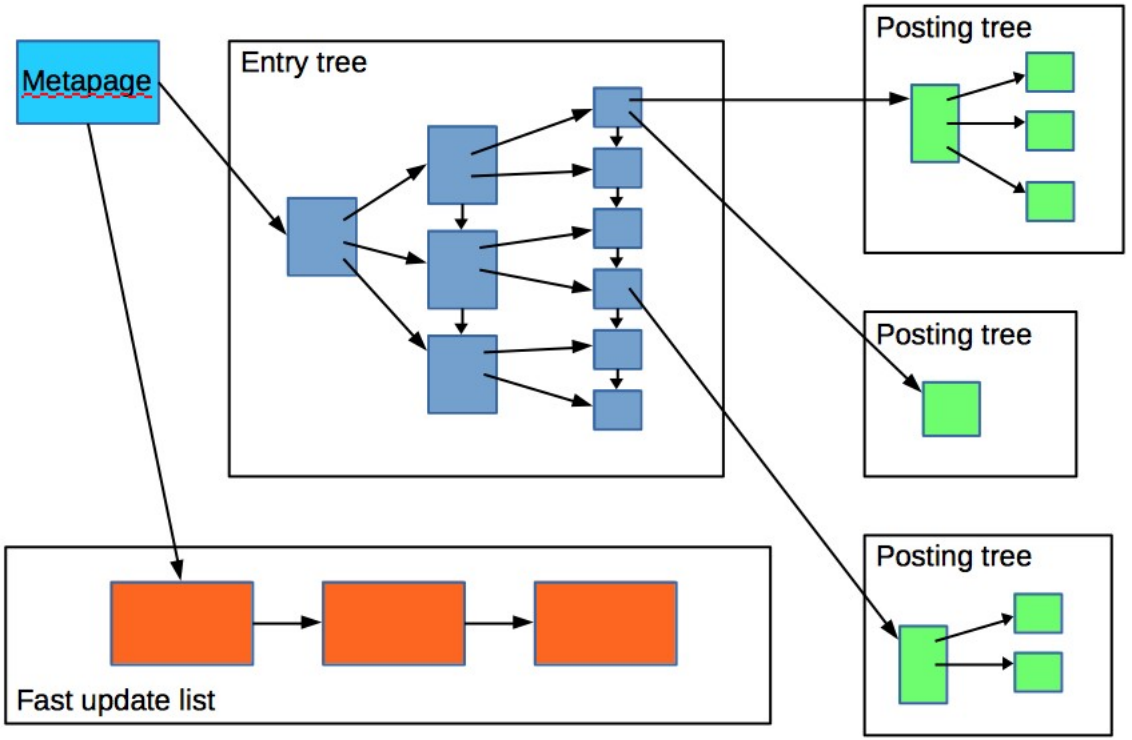
B

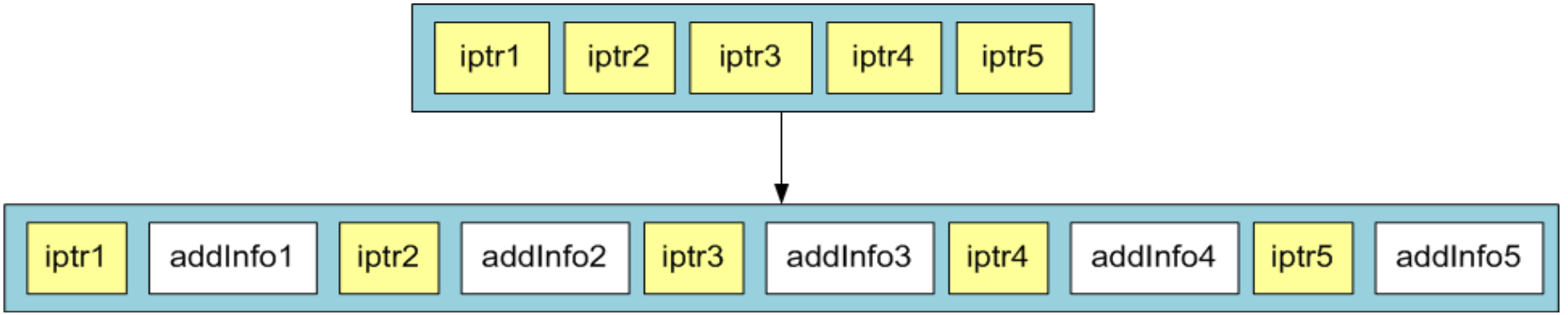
backward wave oscillators, 45

compensation, 30, 68
compressive strength, 54
compressors, 29
computational fluid dynamics, 23, 29
computer games, 56
concurrent engineering, 14
contact resistance, 47, 66
convertors, 22
coplanar waveguide components, 40
Couette flow, 21
creep, 17
crystallisation, 64



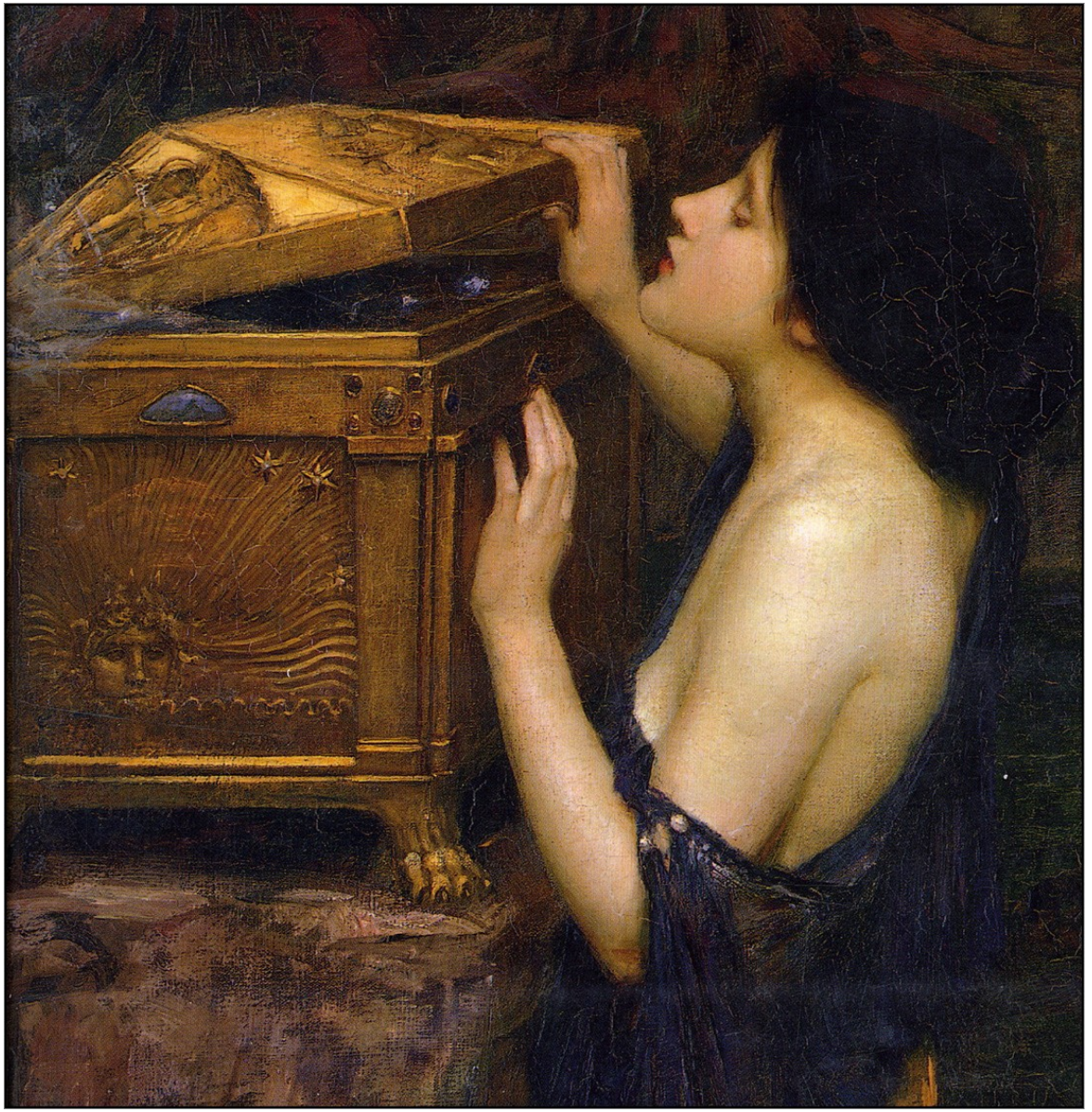
Улучшим GIN


Metapage
Entry tree
Posting tree
Posting tree
Posting tree
Fast update list


iptr1
iptr2
iptr3
iptr4
iptr5
iptr1
addInfo1
iptr2
addInfo2
iptr3
addInfo3
iptr4
addInfo4
iptr5
addInfo5

9.6 opens «Pandora box»

Create access methods as extension ! Let's call it RUM



CREATE INDEX ... USING RUM

- Ранжируем прямо в индексе
- Изобретение: tsvector <-> tsquery

```
CREATE INDEX ti2_rum_fts_idx ON ti2 USING rum(text_vector rum_tsvector_ops);

SELECT docid, ts_rank(text_vector, to_tsquery('english', 'title')) AS rank
FROM ti2
WHERE text_vector @@ to_tsquery('english', 'title')
ORDER BY
text_vector <-> plainto_tsquery('english','title') LIMIT 3;
                                        QUERY PLAN
-------------------------------------------------------------------------------------------
 Limit (actual time=13.843..13.884 rows=3 loops=1)
   ->  Index Scan using ti2_rum_fts_idx on ti2 (actual time=13.841..13.881 rows=3 loops=1)
         Index Cond: (text_vector @@ '''titl'''::tsquery)
         Order By: (text_vector <-> '''titl'''::tsquery)
 Planning time: 0.134 ms
 Execution time: 14.030 ms vs 433 ms !
(6 rows)
```

CREATE INDEX ... USING RUM

- Top-10 (out of 222813) postings with «Tom Lane»
  - GIN index — 1374.772 ms

```
SELECT subject, ts_rank(fts,plainto_tsquery('english', 'tom lane')) AS rank
FROM pglist WHERE fts @@ plainto_tsquery('english', 'tom lane')
ORDER BY rank DESC LIMIT 10;
                                                        QUERY PLAN
-----------------------------------------------------------------------------------------------------------
 Limit (actual time=1374.277..1374.278 rows=10 loops=1)
   ->  Sort (actual time=1374.276..1374.276 rows=10 loops=1)
         Sort Key: (ts_rank(fts, '''tom'' & ''lane'''::tsquery)) DESC
         Sort Method: top-N heapsort  Memory: 25kB
         ->  Bitmap Heap Scan on pglist (actual time=98.413..1330.994 rows=222813 loops=1)
               Recheck Cond: (fts @@ '''tom'' & ''lane'''::tsquery)
               Heap Blocks: exact=105992
               ->  Bitmap Index Scan on pglist_gin_idx (actual time=65.712..65.712
rows=222813 loops=1)
                     Index Cond: (fts @@ '''tom'' & ''lane'''::tsquery)
 Planning time: 0.287 ms
 Execution time: 1374.772 ms
(11 rows)
```

CREATE INDEX ... USING RUM

- Top-10 (out of 222813) postings with «Tom Lane»
  - RUM index — 102.161 ms vs 1374.772 ms !!!

```
create index pglist_rum_fts_idx on pglist using rum(fts rum_tsvector_ops);

SELECT subject  FROM pglist WHERE fts @@ plainto_tsquery('tom lane')
ORDER BY fts <-> plainto_tsquery('tom lane') LIMIT 10;
                                                QUERY PLAN
-----------------------------------------------------------------------------------------
 Limit (actual time=101.381..101.486 rows=10 loops=1)
   ->  Index Scan using pglist_rum_fts_idx on pglist (actual time=101.380..101.485
rows=10 loops=1)
         Index Cond: (fts @@ plainto_tsquery('tom lane'::text))
         Order By: (fts <-> plainto_tsquery('tom lane'::text))
 Planning time: 0.282 ms
 Execution time: 102.161 ms
(6 rows)
```

6.7 mln classifieds

	GIN	RUM	Functional RUM	Sphinx
Table size	6.0 GB	6.0 GB	2.87 GB	-
Index size	0.69 GB	1.27 GB	1.27 GB	1.12 GB
Index build time	216 sec	303 sec	718sec	180 sec
Queries in 8h	**3.0 mln.**	**42.7 mln.**	**42.7 mln.**	**61.0 mln.**

Inverse FTS (FQS)

Найти запросы, которым удовлетворяет заданный текст

```
SELECT * FROM queries;
                 q                  |  tag
------------------------------------+-------
 'supernova' & 'star'               | sn
 'black'                            | color
 'big' & 'bang' & 'black' & 'hole'  | bang
 'spiral' & 'galaxi'                | shape
 'black' & 'hole'                   | color
(5 rows)


SELECT * FROM queries WHERE
 to_tsvector('black holes never exists before we think about them')
@@ q;
        q          |  tag
-------------------+-------
 'black'           | color
 'black' & 'hole'  | color
(2 rows)
```

Inverse FTS (FQS)

- Поддерживается RUM – сохраняем ветки запроса в допинфо

Find queries for the first message in postgres mailing lists

```
\d pg_query
   Table "public.pg_query"
 Column |  Type   | Modifiers
--------+---------+-----------
 q      | tsquery |
 count  | integer |
Indexes:
    "pg_query_rum_idx" rum (q)                  33818 queries


select q from pg_query pgq, pglist where q @@ pglist.fts and pglist.id=1;
             q
--------------------------
 'one' & 'one'
 'postgresql' & 'freebsd'
(2 rows)
```

Inverse FTS (FQS)

```
create index pg_query_rum_idx on pg_query using rum(q);
select q from pg_query pgq, pglist where q @@ pglist.fts and pglist.id=1;
                                                QUERY PLAN
-----------------------------------------------------------------------------------------------
 Nested Loop (actual time=0.719..0.721 rows=2 loops=1)
   ->  Index Scan using pglist_id_idx on pglist
(actual time=0.013..0.013 rows=1 loops=1)
         Index Cond: (id = 1)
   ->  Bitmap Heap Scan on pg_query pgq
(actual time=0.702..0.704 rows=2 loops=1)
         Recheck Cond: (q @@ pglist.fts)
         Heap Blocks: exact=2
         ->  Bitmap Index Scan on pg_query_rum_idx
(actual time=0.699..0.699 rows=2 loops=1)
               Index Cond: (q @@ pglist.fts)
 Planning time: 0.212 ms
 Execution time: 0.759 ms
(10 rows)
```

Inverse FTS (FQS)

```
select id, t.subject, count(*) as cnt into pglist_q  from pg_query,
(select id, fts, subject from pglist) t where t.fts @@ q
group by id, subject order by cnt desc limit 1000;

select * from pglist_q  order by cnt desc limit 5;
   id   |                      subject                       | cnt
--------+----------------------------------------------------+------
 248443 | Packages patch                                     | 4472
 282668 | Re: release.sgml, minor pg_autovacuum changes      | 4184
 282512 | Re: release.sgml, minor pg_autovacuum changes      | 4151
 282481 | release.sgml, minor pg_autovacuum changes          | 4104
 243465 | Re: [HACKERS] Re: Release notes                    | 3989
(5 rows))
```

Поиск фраз (2008!)

- Запросы 'A & B'::tsquery и 'B & A'::tsquery эквивалентны
- Поиск фраз – учитывает порядок слов в запросе

  Ресзультаты 'A B' и 'B A' должны быть разные!
- Новый оператор: PHRASE (<->):
  - Порядок
  - Расстояние

a <n> b == a & b & (∃ i,j : pos(b)i – pos(a)j = n)

A <n> B - A «phrase n» B

Определение

- Оператор может возвращать:
  - false
  - true и массив позиция ПРАВОГО аргумента, которые удовлетворяют условию расстояния.

$$A <-> B \neq B <-> A$$

Свойства

- 'A <n> B <m> C' → '(A <n> B) <m> C' →
  длина фразы = n + m
- 'A <n> (B <m> C)' →
  длина фразы = min(n, m)
  И да, B может встретиться раньше A
- 'A <0> B' - “угловой” случай.

Phrase search - example

- `TSQUERY phraseto_tsquery([CFG,] TEXT)`

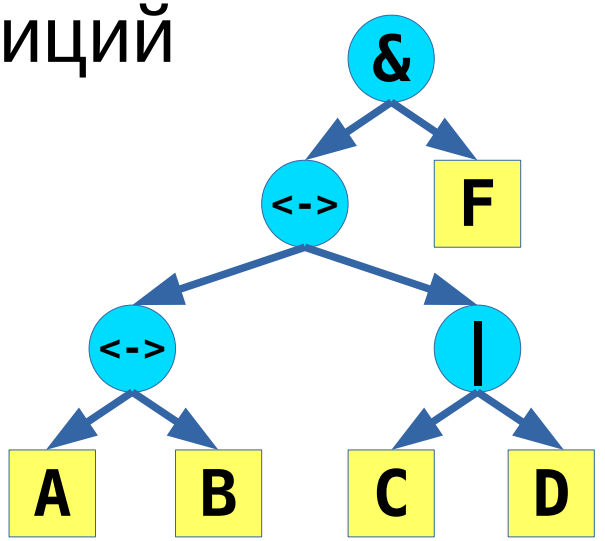
```
select phraseto_tsquery('english','PostgreSQL can be extended by the user in many ways');
                    phraseto_tsquery
-----------------------------------------------------------------
 ( ( ( 'postgresql' <3> 'extend' ) <3> 'user' ) <2> 'mani' ) <-> 'way'
(1 row)
```

Стоп-слова опускаются!

Поиск фраз - внутренности

- Поиск фраз имеет дополнительную сложность, связанную с вычислением позиций

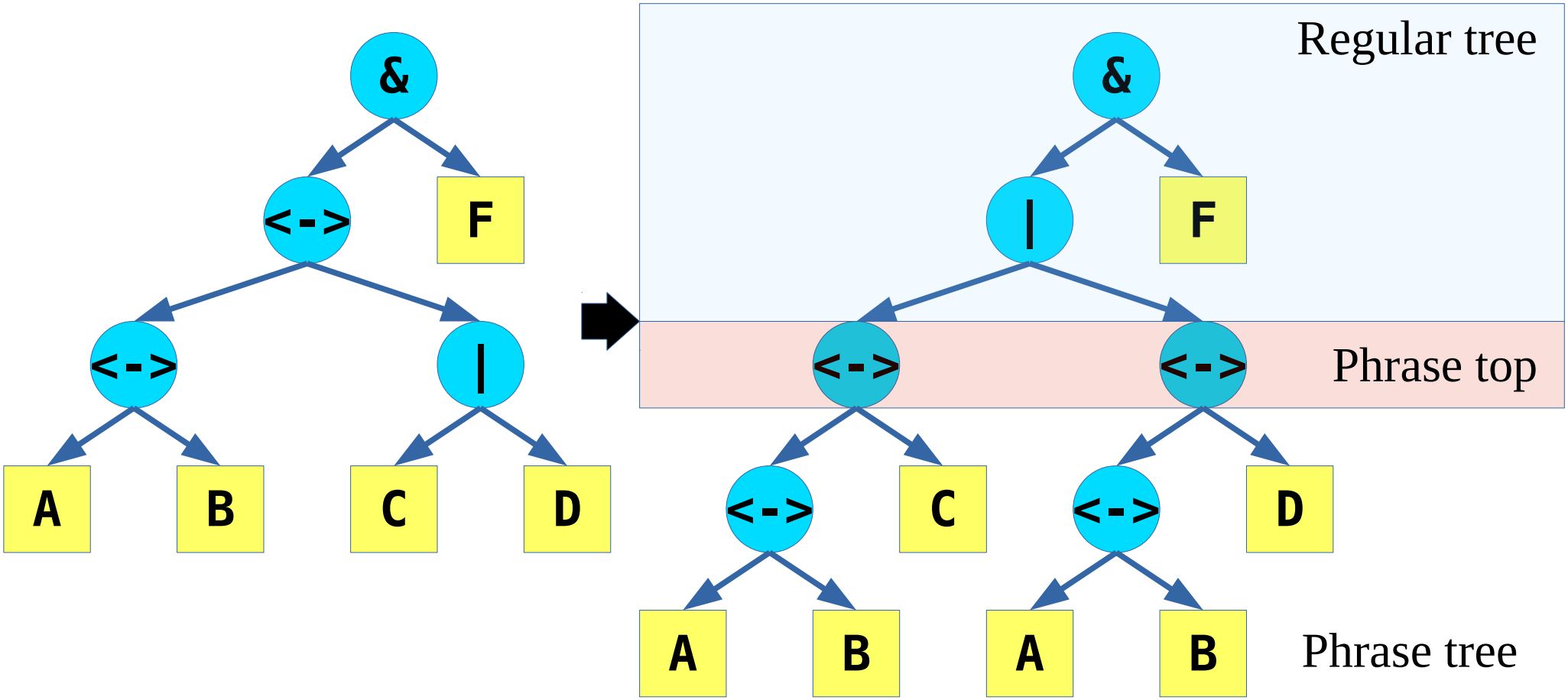
```
( (A <-> B) <-> (C | D) ) & F
```



- Хотелось бы избежать этих сложностей для обычных операторов
- Модифицируем запрос так, что бы все фразовые операторы были в «глубине»

Трасформатор

```
( (A <-> B) <-> (C | D) ) & F
```

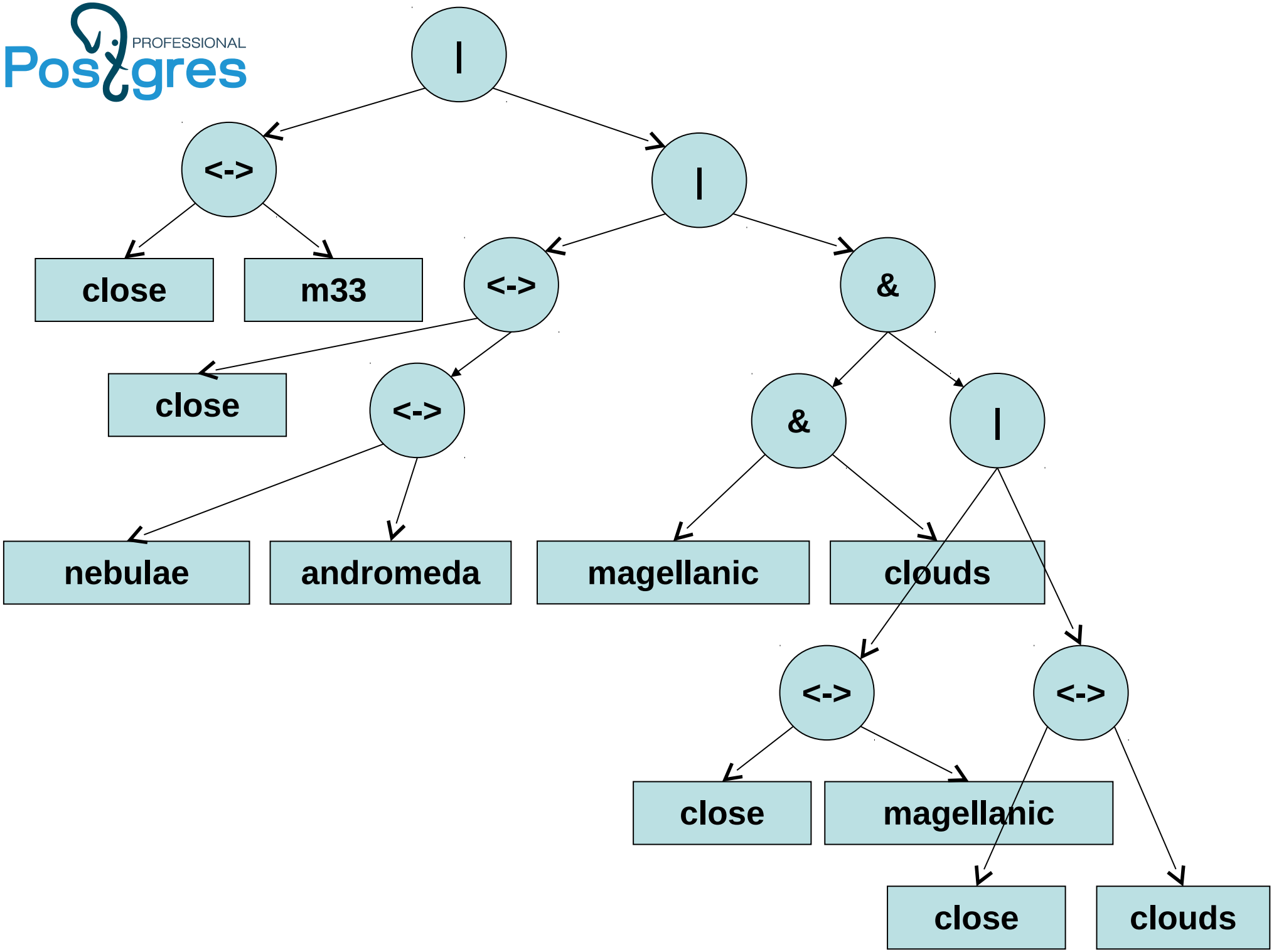


Example

Query: close <-> galaxies

After dictionary: close <-> (m33 |
(andromeda <-> nebulae | (magellanic & clouds))

Phrase: close <-> m33 |
(close <-> (andromeda <-> nebulae)) |
(magellanic & clouds &
(close <-> magellanic |
close <-> clouds
)
)


|
<->
|
close
m33
<->
&
close
<->
&
|
nebulae
andromeda
magellanic
clouds
<->
<->
close
magellanic
close
clouds

Трасформатор

```
a <-> (b&c) => (a<->b) & (a<->c)
(a&b) <-> c => (a<->c) & (b<->c)
a <-> (b|c) => (a<->b) | (a<->c)
(a|b) <-> c => (a<->c) | (b<->c)
a <-> !b    => a & !(a<->b)
```

Отсутствуют такие вхождения A, после которых есть B

```
!a <-> b    => !(a<->b) & b
```

Отсутствуют такие вхождения B, перед которыми есть A

Трансформатор

```
# select '( A | B ) <-> ( D | C )'::tsquery;
                          tsquery
  ---------------------------------------------
'A' <-> 'D' | 'B' <-> 'D' | 'A' <-> 'C' | 'B' <-> 'C'

# select 'A <-> ( B & ( C | ! D ) )'::tsquery;
                                tsquery
-----------------------------------------------------------
('A' <-> 'B') & ( 'A' <-> 'C' | 'A' & !( 'A' <-> 'D' ) )
```

Пример

- 1.1 mln postings (postgres mailing lists)

select count(*) from pglist where fts @@ to_tsquery('english','tom <-> lane');
 count

 222777
(1 row)

Sequential Scan (1.7 s (<->) vs 1.6 s (&)):

```
select count(*) from pglist where fts @@ to_tsquery('english', 'tom <-> lane');
                                                    QUERY PLAN
------------------------------------------------------------------------------------------------------------------
 Finalize Aggregate (actual time=1700.280..1700.280 rows=1 loops=1)
   ->  Gather (actual time=1700.228..1700.277 rows=3 loops=1)
         Workers Planned: 2
         Workers Launched: 2
         ->  Partial Aggregate (actual time=1696.119..1696.119 rows=1 loops=3)
               ->  Parallel Seq Scan on pglist (actual time=2.356..1683.499 rows=74259
loops=3)
                     Filter: (fts @@ '''tom'' <-> ''lane'''::tsquery)
                     Rows Removed by Filter: 263664
 Planning time: 0.270 ms
 Execution time: 1709.092 ms
(10 rows)
```

Пример

- 1.1 mln postings (postgres mailing lists)

select count(*) from pglist where fts @@ to_tsquery('english','tom <-> lane');
 count

 222777
(1 row)

GIN index (1.1 s (<->) vs 0.48 s (&)): Use recheck, phrase is slow vs fts

```
select count(*) from pglist where fts @@ to_tsquery('english', 'tom <-> lane');
                                                    QUERY PLAN
------------------------------------------------------------------------------------------------------------------
 Aggregate (actual time=1074.983..1074.984 rows=1 loops=1)
   ->  Bitmap Heap Scan on pglist (actual time=84.424..1055.770 rows=222777 loops=1)
         Recheck Cond: (fts @@ '''tom'' <-> ''lane'''::tsquery)
         Rows Removed by Index Recheck: 36
         Heap Blocks: exact=105992
         ->  Bitmap Index Scan on pglist_gin_idx (actual time=53.628..53.628 rows=222813
loops=1)
               Index Cond: (fts @@ '''tom'' <-> ''lane'''::tsquery)
 Planning time: 0.329 ms
 Execution time: 1075.157 ms
(9 rows)
```

Пример

- 1.1 mln postings (postgres mailing lists)

select count(*) from pglist where fts @@ to_tsquery('english', 'tom <-> lane');
 count

 222777
(1 row)

RUM index (0.5 s (<-> vs 0.48 s (&)): Use positions in addinfo, no overhead of phrase search !

```
select count(*) from pglist where fts @@ to_tsquery('english', tom <-> lane');
                                                    QUERY PLAN
------------------------------------------------------------------------------------------------------------------
 Aggregate (actual time=513.517..513.517 rows=1 loops=1)
   ->  Bitmap Heap Scan on pglist (actual time=134.109..497.814 rows=221919 loops=1)
         Recheck Cond: (fts @@ to_tsquery('tom <-> lane'::text))
         Heap Blocks: exact=105509
         ->  Bitmap Index Scan on pglist_rum_fts_idx (actual time=98.746..98.746
rows=221919 loops=1)
               Index Cond: (fts @@ to_tsquery('tom <-> lane'::text))
 Planning time: 0.223 ms
 Execution time: 515.004 ms
(8 rows)
```

Some FTS problems #3

Как скомбинировать полнотекстовый поиск и, скажем, дату документа?

- Добавим ее в допинфо в индексе!

```
create index pglist_tsvtime_rum_idx on pglist using rum(fts
rum_tsvector_timestamp_ops, sent) WITH (orderby = 'sent', addto =
'fts');

select sent,  subject from pglist
where fts @@ to_tsquery('tom & lane')
order by sent <=> '2000-01-01'::timestamp limit 5;
                                                    QUERY PLAN
-------------------------------------------------------------------------------
 Limit (actual time=89.499..89.505 rows=5 loops=1)
   ->  Index Scan using pglist_tsvtime_rum_idx on pglist (actual
time=89.498..89.504 rows=5 loops=1)
         Index Cond: (fts @@ to_tsquery('tom & lane'::text))
         Order By: (sent <=> '2000-01-01 00:00:00'::timestamp without
time zone)
 Planning time: 0.286 ms
 Execution time: 90.317 ms vs 545 ms !
```

RUM Todo

- <=>, <=|, |=> без сортировки в индексе
- Позиционка И дата, а не ИЛИ
- Добавить поддержку anyarray/anyint
- Улучшить сортировку (TF/IDF)
- 9.6+ only (WIP)
  https://github.com/postgrespro/rum

Словрь каак расширение

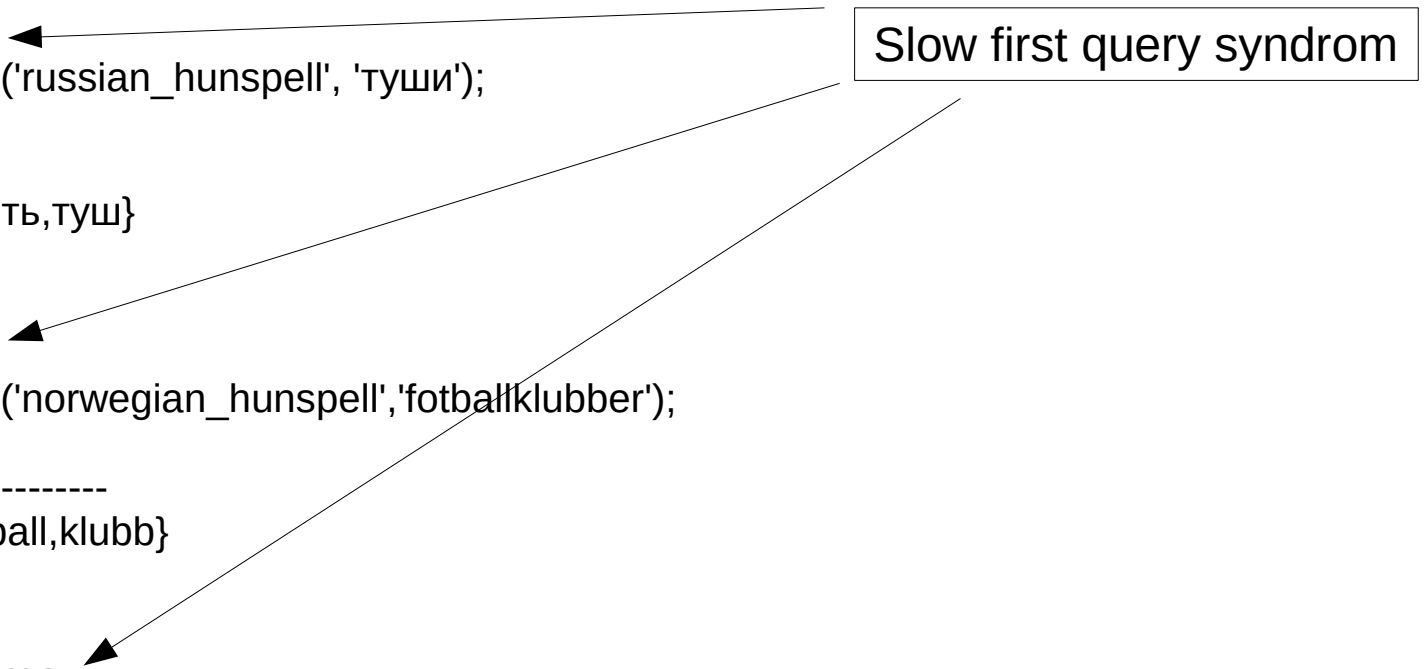
- Now it's easy (Artur Zakirov, PostgresPro)
  https://github.com/postgrespro/hunspell_dicts

```
CREATE EXTENSION hunspell_ru_ru; -- creates russian_hunspell dictionary
CREATE EXTENSION hunspell_en_us; -- creates english_hunspell dictionary
CREATE EXTENSION hunspell_nn_no; -- creates norwegian_hunspell dictionary
SELECT ts_lexize('english_hunspell', 'evening');
   ts_lexize
----------------
 {evening,even}
(1 row)

Time: 57.612 ms
SELECT ts_lexize('russian_hunspell', 'туши');
       ts_lexize
------------------------
 {туша,тушь,тушить,туш}
(1 row)

Time: 382.221 ms
SELECT ts_lexize('norwegian_hunspell','fotballklubber');
          ts_lexize
-------------------------------
 {fotball,klubb,fot,ball,klubb}
(1 row)

Time: 323.046 ms
```



Предзагруженные словари

- Now it's easy (Artur Zakirov, PostgresPro + Thomas Vondra)
  https://github.com/postgrespro/shared_ispell

```
CREATE EXTENSION shared_ispell;
CREATE TEXT SEARCH DICTIONARY english_shared (
    TEMPLATE = shared_ispell,
    DictFile = en_us,
    AffFile = en_us,
    StopWords = english
);
CREATE TEXT SEARCH DICTIONARY russian_shared (
    TEMPLATE = shared_ispell,
    DictFile = ru_ru,
    AffFile = ru_ru,
    StopWords = russian
);
time for i in {1..10}; do echo $i; psql postgres -c "select ts_lexize('russian_shared', 'туши')" > /dev/null;  done
1
2
…..
10
```

```
real  0m0.170s          real 0m3.809s
user  0m0.015s    VS    user0m0.015s
sys   0m0.027s          sys  0m0.029s
```

Редактирование tsvector

- Stas Kelvich (PostgresPro)
- setweight(tsvector, «char», text[]) - простановка меток лексемам

```
select setweight( to_tsvector('english', '20-th anniversary of PostgreSQL'),
'A',  '{postgresql,20}');
                  setweight
-----------------------------------------------
 '20':1A 'anniversari':3 'postgresql':5A 'th':2
(1 row)
```

- ts_delete(tsvector, text[]) - удаление лексем

```
select ts_delete( to_tsvector('english', '20-th anniversary of PostgreSQL'),
'{20,postgresql}'::text[]);
       ts_delete
------------------------
 'anniversari':3 'th':2
(1 row)
```

Tsvector editing functions

- unnest(tsvector)

```
select * from unnest( setweight( to_tsvector('english',
'20-th anniversary of PostgreSQL'),'A',  '{postgresql,20}'));
   lexeme    | positions | weights
-------------+-----------+---------
 20          | {1}       | {A}
 anniversari | {3}       | {D}
 postgresql  | {5}       | {A}
 th          | {2}       | {D}
(4 rows)
```

- tsvector_to_array(tsvector) — tsvector to text[]
  array_to_tsvector(text[])

```
select tsvector_to_array( to_tsvector('english',
'20-th anniversary of PostgreSQL'));
       tsvector_to_array
-------------------------------
 {20,anniversari,postgresql,th}
(1 row)
```

Tsvector editing functions

- ts_filter(tsvector,text[]) - фильтрация лексем по метке

```
select ts_filter($$'20':2A 'anniversari':4C 'postgresql':1A,6A 'th':3$$::tsvector,
'{C}');
    ts_filter
------------------
 'anniversari':4C
(1 row)

select ts_filter($$'20':2A 'anniversari':4C 'postgresql':1A,6A 'th':3$$::tsvector,
'{C,A}');
                   ts_filter
-----------------------------------------------
 '20':2A 'anniversari':4C 'postgresql':1A,6A
(1 row)
```

PROFESSIONAL Postgres

www.postgrespro.ru