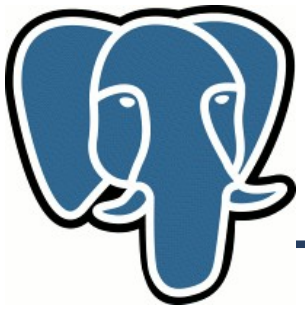
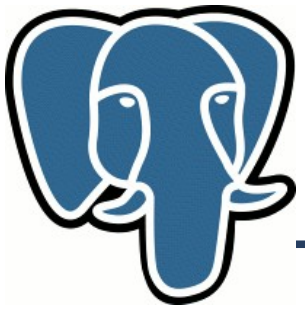


Some recent advances in full-text search

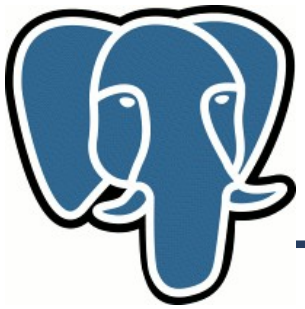
Oleg Bartunov, Teodor Sigaev

Sternberg Astronomical Institute,
Moscow State University, Russia



Talk roadmap

- Full-text search introduction
- Main topics
  - Phrase Search
  - Dictionaries API
- New features (already in 8.4)
- Future features
- Tips and Tricks



Full-text search in PostgreSQL

=# select **'a fat cat sat on a mat and ate a fat rat'::tsvector**

@@

'cat & rat':: tsquery;

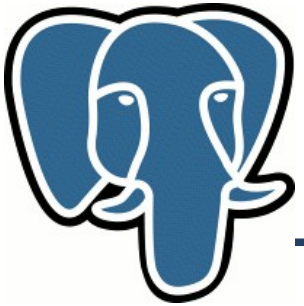
- **tsvector** – storage for document
- – sorted array of lexemes with optional positional and weight information
- **tsquery** – textual data type for query
- – Boolean operators - & | ! ()

'telefonsvarer' =>
'telefonsvarer' | 'telefon' & 'svar'

- **FTS operator**

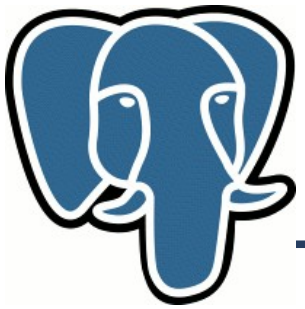
  tsvector @@ tsquery

- to_tsvector, to_tsquery, plainto_tsquery
- Indexes: GiST, GIN



Talk roadmap

- Full-text search introduction
- Main topics
  - **Phrase Search**
  - Dictionaries API
- New features (already in 8.4)
- Future features
- Tips and Tricks



Phrase search - definition

A $ B: word 'A' followed by 'B':

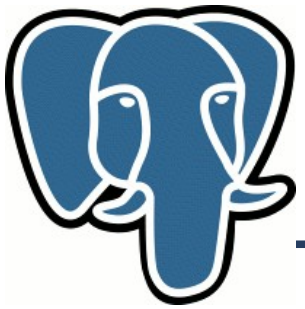
- A & B (the same priority)
- exists at least one pair of positions P_B, P_A , so that $0 \le P_B - P_A \le 1$ (distance condition)

A $[n] B: $0 \le P_B - P_A \le n$

Result of operation:

- false
- true and array of positions of left argument which satisfy distance condition (without positional information $ is equivalent to &)

$ is very similar to & except: A $ B ≠ B $ A



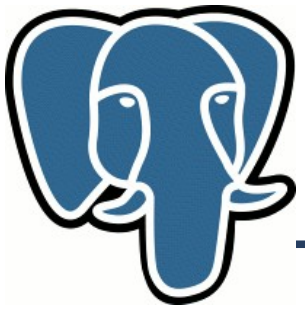
Phrase search - properties

'A $[n] B $[m] C' → '(A $[n] B) $[m] C' →
matched phrase length ≤ max(n, m)
Note: 'A C B' matched by '(A $[2] B) $ C'

'A $[n] (B $[m] C') →
matched phrase length ≤ n + m
Note: Order is preserved for any n, m

'A $[0] B' matches the word with two different forms (infinitives)

```
=# SELECT ts_lexize('ispell','bookings');
   ts_lexize
---------------
 {booking,book}
to_tsvector('bookings') @@ 'booking $[0] book'::tsquery
```



Phrase search - practice

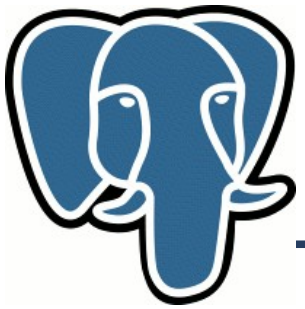
Phrase:

- 'A B C' → 'A $ (B $ C)'
- 'A B C' → '(A $ B) $[2] C'
- TSQUERY phraseto_tsearch([CFG,] TEXT)

Stop-words: 'A the B' → 'A $[2] B'

What shall we do with complex queries?

A $ (B & (C | ! D) → ???



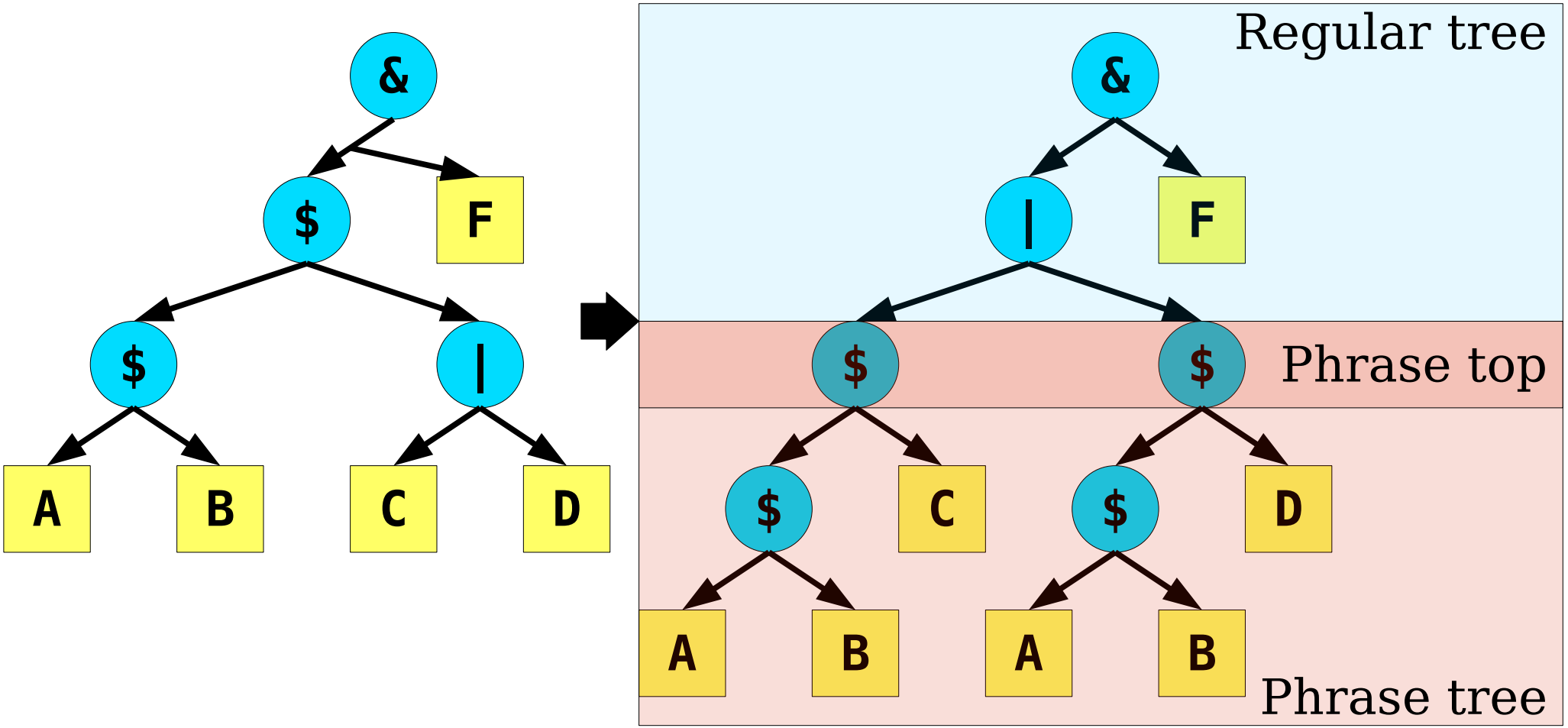
Phrase search - internals

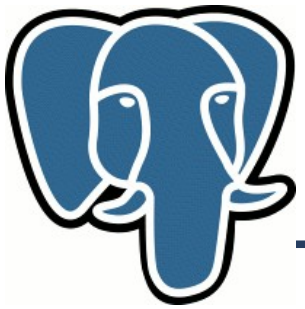
Phrase search has overhead, since it requires access and operations on posting lists

To avoid slowdown of existing tsearch, executor of tsquery should not access positions without necessity. To facilitate this, any $ operations pushed down in query tree, so tsearch executor can call special phrase executor for the top $ operation, which will work only with query tree containing only $ operations.

Phrase search - transformation

```
( (A $ B) $ (C | D) ) & F
```





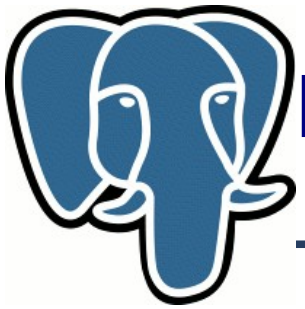
Phrase search - push down

```
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)
```

there is no position of A followed by B

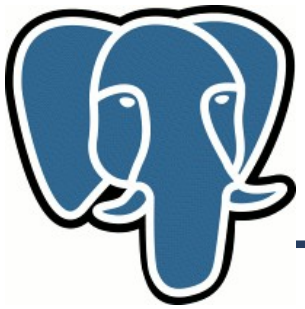
```
!a $ b    => b & !(a$b)
```

there is no position of B precedenced by A



Phrase search - transformation

```
# 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' ) )
```



Phrase search - example

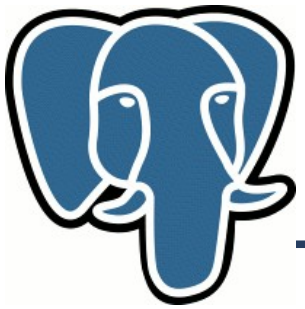
'PostgreSQL can be extended by the user in many ways' ->

```
# select phraseto_tsquery('PostgreSQL can be extended
                           by the user in many ways');
                              phraseto_tsquery
  ------------------------------------------------------------------

'postgresql' $[3] ( 'extend' $[3] ( 'user' $[2] ( 'mani' $ 'way' ) ) )
```

Can be written by hand:

```
'postgresql' $[3] extend $[6] user $[8] mani $[9] way
```

Difficult to modify, use phraseto_tsquery() function !



Phrase search - TODO

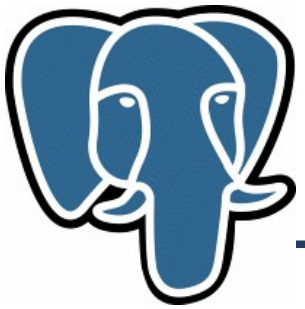
Ranking functions

Headline generation

Rewrite subsystem

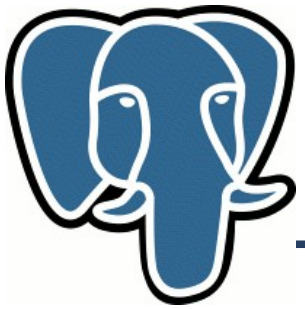
Concatenation of two tsquery by $ operation: $$?
- like other concatenations: &&, || and !!
- distance $$[2] !, functional interface ?

Need testing for agglutinative languages (norwegian, german, etc)



Talk roadmap

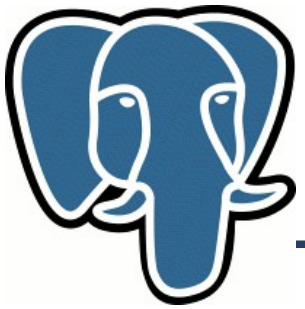
- Full-text search introduction
- Main topics
  - Phrase Search
  - **Dictionaries API**
- New features (already in 8.4)
- Future features
- Tips and Tricks



Dictionaries

Lexeme's type	Dict #1	Dict #2	Dict #N
asciiword	synonym	en_ispell	en_stem
int	simple		
float	real_dict		

```
# \dF+ english
Text search configuration "pg_catalog.english"
Parser: "pg_catalog.default"
      Token       | Dictionaries
------------------+--------------
 asciihword       | english_stem
 asciiword        | english_stem
 email            | simple
 file             | simple
....
```



Dictionaries - examples

Integers

'123456789' -> '123456'

Roman numbers

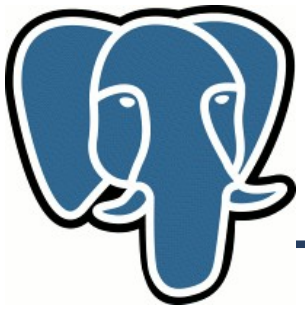
'XIX' -> '19'

Colours

'FFFFFF' -> 'white'

Regexp

H(\s|-)?(alpha|beta|gamma) h$2 — spectral lines of hydrogen

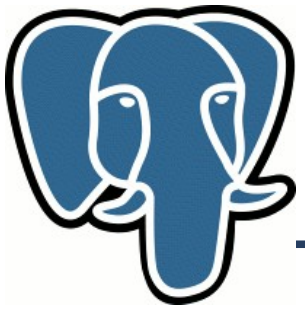


Dictionaries - interface

```
void* dictInit(List *dictoptions)
```

- list of dictoptions actually contains list of DefElem structures (see headers)
- returns pointer to the palloc'ed dictionary structure
- Can be expensive (ispell)

```
TSLexeme* dictLexize(
    void* dictData, // returned by dictInit()
    char* lexeme,   // not zero-terminated
    int   lenlexeme,
    DictSubState *substate  // optional
);
```



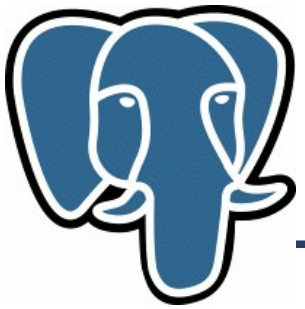
Dictionaries – output

```
typedef struct {
        uint16      nvariant; // optional
        uint16      flags;    // optional
        char        *lexeme;
  } TSLexeme;
```

dictLexize returns NULL – dictionary doesn't recognize the lexeme

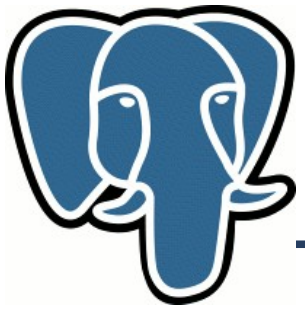
dictLexize returns array of TSLexeme (last element TSLexeme->lexeme is NULL)

dictLexize returns empty array – dictionary recognizes the lexeme, but it's a stop-word



Dictionaries - output

```
SELECT ts_lexize('en_ispell','bookings');


TSLexeme array:
#  nvariant flags  lexeme
0       0       0    booking
1       0       0    book
2       0       0    NULL
```



Agglutinative Languages

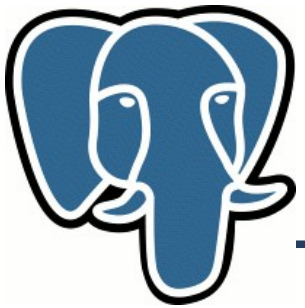
German, norwegian, ...
http://en.wikipedia.org/wiki/Agglutinative_language

Concatenation of words without space

Query - Fotballklubber
Document - Klubb on fotballfield

How to find document ?

Split words and build search query
'fotbalklubber' =>
' (fotball & klubb) | (fot & ball & klubb) '

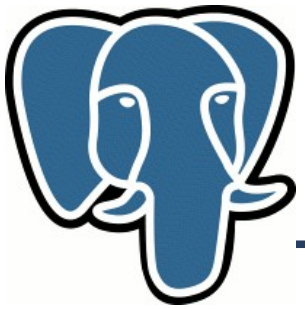


Dictionaries - TSLexeme->nvariant

Agglutinative languages have several variants of word's splitting:
Word 'foobarcom' (imaginary)

Lexeme	nvariant	Comment
foo	1	
bar	1	
com	1	
foob	2	-a- is an affix (interfix)
rcom	2	

```
tsvector: 'bar:1 com:1 foo:1 foob:1 rcom:1'
tsquery: '(foob & rcom) | (foo & bar & com)'
```



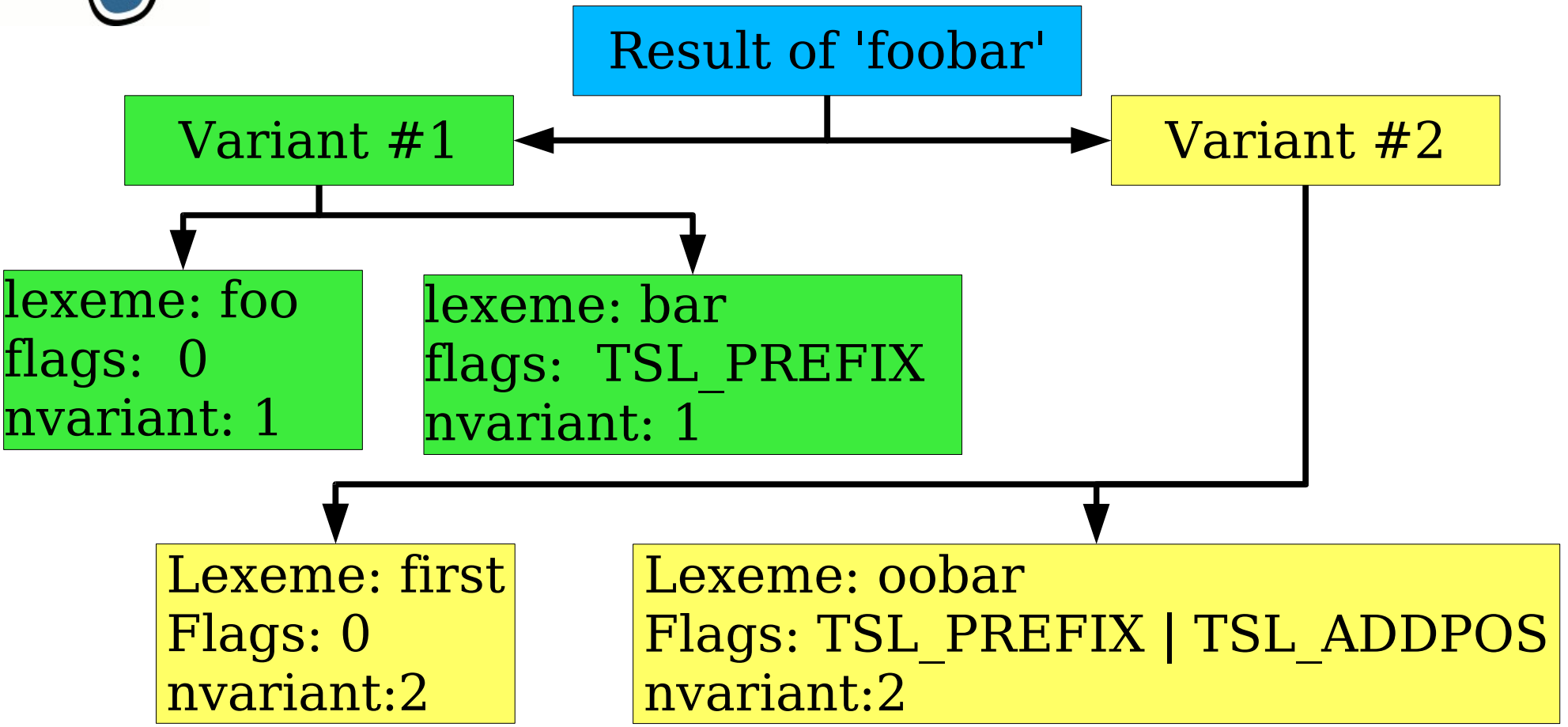
Dictionaries - output

Each TSLexeme describes one normalized lexeme

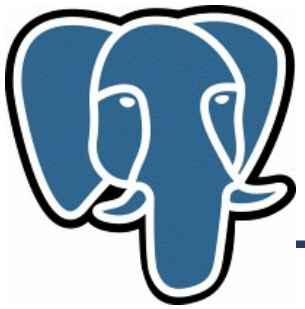
TSLexeme->flags is an OR-ed:

- TSL_PREFIX indicates to use prefix search for this lexeme
  Note: dictionaries are planned for 8.5
- TSL_ADDPOS points to parser to increase position's counter
  Note: currently only thesaurus dictionary uses it

Dictionaries – output (imaginary)



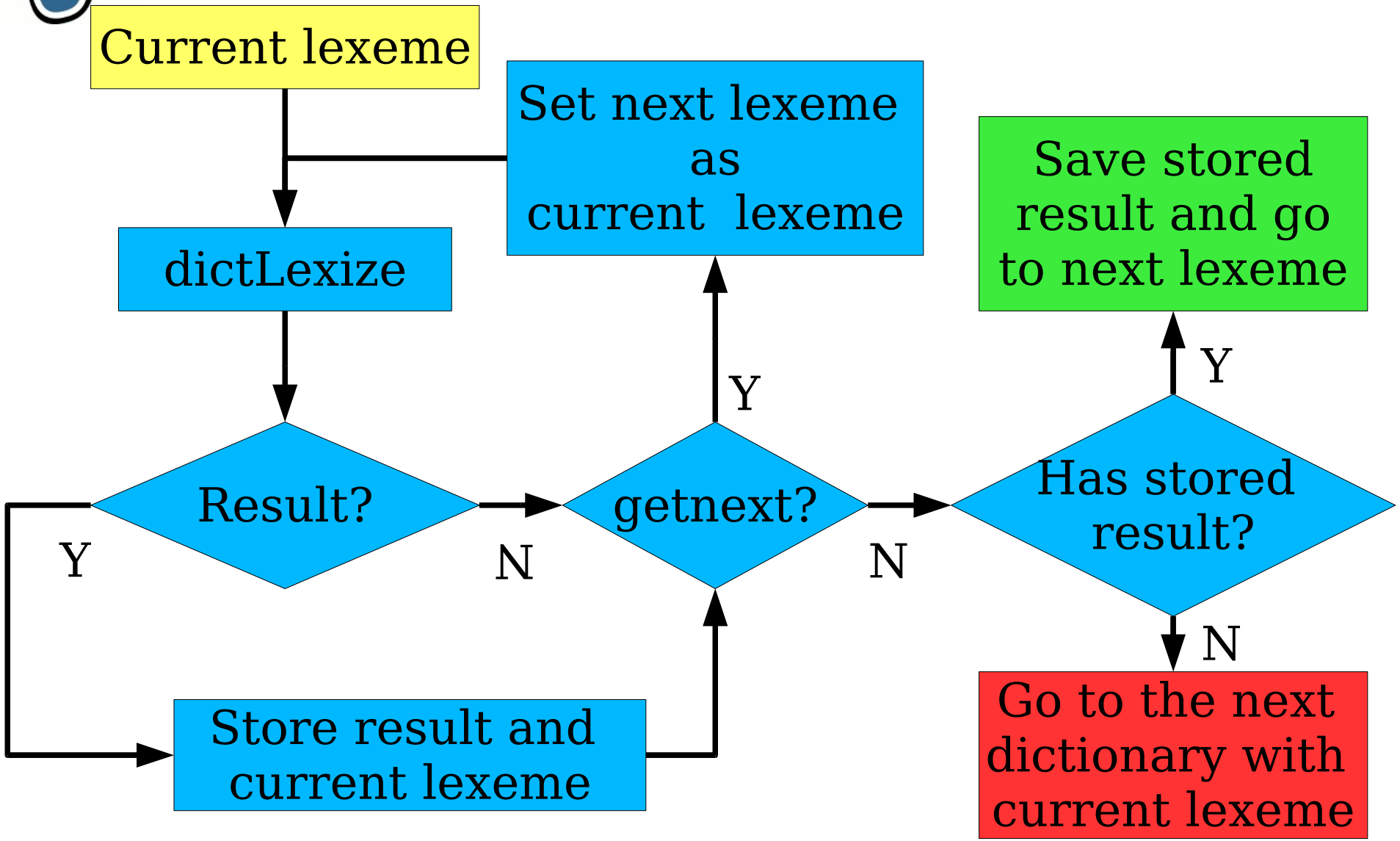

```
tsvector: 'foo:1 bar:1 first:1 oobar:2'
tsquery: '(foo & bar:*) | (first & oobar:*)'
```

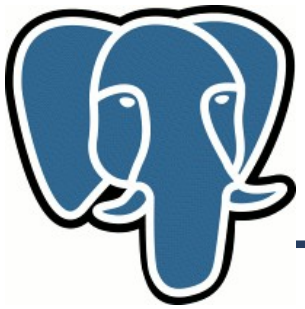


Dictionaries - several words

```
typedef struct {
      bool          isend;   // in: marks end of text
                             // (input lexeme is invalid!)
      bool          getnext; // out: dictionary asks for
                             // a next lexeme
      void         *private; // internal state of
                             // dictionary while it's
                             // asking a next lexeme
  } DictSubState;
```

Dictionaries – several words

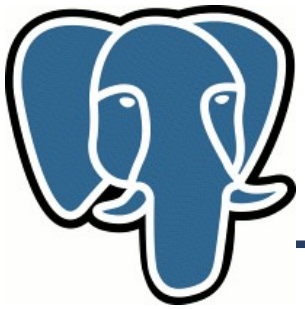




Dictionaries – filter for 8.5

New TSLexeme->flags: TSL_FILTER

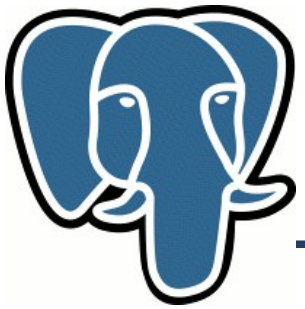
If dictionary returns only one lexeme with TSL_FILTER flag, then that lexeme will be used as an input for the subsequent dictionaries in the chain.



Filter dictionary – unaccent (8.5)

contrib/unaccent provides unaccent text search dictionary and function to remove accents (suffix tree, ~ 25x faster *translate*() solution)

```
1. Unaccent dictionary does nothing and returns NULL.
(lexeme 'Hotels' will be passed to the next dictionary if any)

=# select ts_lexize('unaccent','Hotels') is NULL;
 ?column?
----------
 t

2. Unaccent dictionary removes accent and returns 'Hotel'.
(lexeme 'Hotel' will be passed to the next dictionary if any)

=# select ts_lexize('unaccent','Hôtel');
 ts_lexize
-----------
 {Hotel}
```

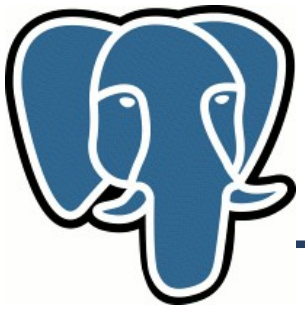


Filter dictionary - unaccent

```
CREATE TEXT SEARCH CONFIGURATION fr ( COPY = french );
ALTER TEXT SEARCH CONFIGURATION fr ALTER MAPPING FOR hword, hword_part, word
    WITH unaccent, french_stem;


=# select to_tsvector('fr','Hôtel de la Mer') @@ to_tsquery('fr','Hotels');
 ?column?
----------
 t
```

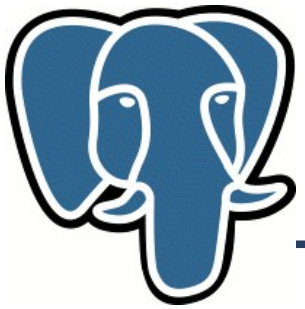
Finally, unaccent dictionary solves the known problem with headline !
(to_tsvector(remove_accent(document)) works with search, but has problem with highlighting)

```
=# select ts_headline('fr','Hôtel de la Mer',to_tsquery('fr','Hotels'));
      ts_headline
------------------------
  <b>Hôtel</b> de la Mer
```



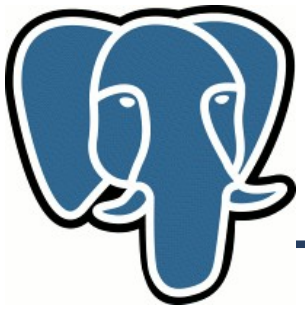
Talk roadmap

- Full-text search introduction
- Main topics
  - Phrase Search
  - Dictionaries API
- New features (already in 8.4)
- Future features
- Tips and Tricks



New features and improvements

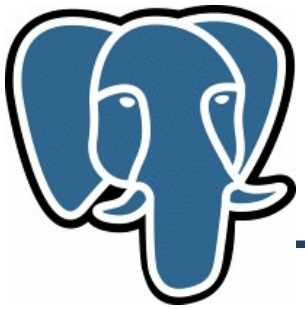
- ts_headline() enhancements (8.4)
- Prefix full-text search support (8.4)
- Devanagari script support (8.4)
- dict_xsyn improvement
- ts_stat() performance improvement (8.4)
- Fast approximated statistics (8.3,8.4)
- GIN improvements: fast update (8.4), partial match support (8.4), multicolumn (8.4)
- contrib/btree_gin (8.4)



ts_headline enhancement

- New parameter **MaxFragments**
  by Sushant Sinha. Default is 0, ts_headline() generates one fragment

```
=# select ts_headline($$
  Text from http://www.postgresql.org/docs/8.3/static/history.html
$$,
 plainto_tsquery('postgresql postgres '), 'MaxFragments=3,
MinWords=3, MaxWords=6');
                              ts_headline
----------------------------------------------------------------------------------
 <b>PostgreSQL</b> is derived from the <b>POSTGRES</b> ...
behind it, <b>PostgreSQL</b> ... <b>PostgreSQL</b> as
"<b>Postgres</b>" (now rarely
```



Prefix full-text search support

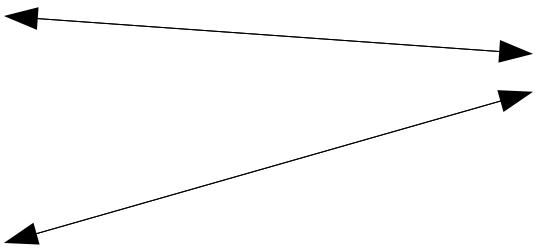
- Prefix full-text search support
  - to_tsquery('supernov:*') - match all documents, which contains words with prefix 'supernov'
  - to_tsquery('supernov:ab*') - the same,but only in titles (weight 'a') and keywords (weight 'b')
  - Can use new GIN partial match feature to speedup search
  - Can be useful if there is no stemmer available

Devanagari script support

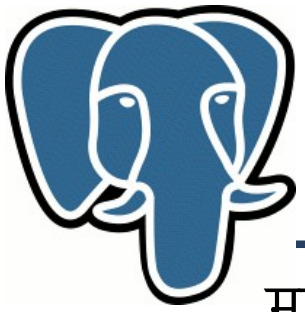
PostgreSQL 8.3- has problem with Devanagari script (http://en.wikipedia.org/wiki/Devanagari - script for Hindi, Marathi, Nepali, Sanscrit,...).

```
select * from ts_parse('default','मदन पुरस्कार पुस्तकालय');
```

```
2     मदन                          Madan Puraskar Pustakalaya
12
2     पुरस
12
2     कार
12
2     पुस
12
2     तकालय
```



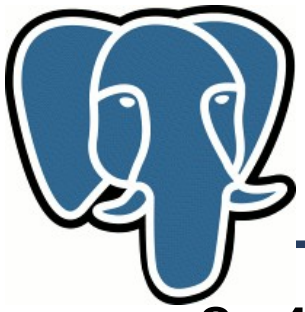
Virama sign (modifier, suppresses inherent vowel) – *punct* in np_NP locale. Breaks all parsers, which use locale.



Devanagari script support

मदन पु रस्कार पु स्तकालय (Madan Puraskar Pustakalaya)

character	byte	UTF-32	encoded as	glyph	name
0	0	00092E	E0 A4 AE	म	DEVANAGARI LETTER MA
1	3	000926	E0 A4 A6	द	DEVANAGARI LETTER DA
2	6	000928	E0 A4 A8	न	DEVANAGARI LETTER NA
3	9	000020	20		SPACE
4	10	00092A	E0 A4 AA	प	DEVANAGARI LETTER PA
5	13	000941	E0 A5 81	ु	DEVANAGARI VOWEL SIGN U
6	16	000930	E0 A4 B0	र ु	DEVANAGARI LETTER RA
7	19	000938	E0 A4 B8	स	DEVANAGARI LETTER SA
8	22	00094D	E0 A5 8D	्	DEVANAGARI SIGN VIRAMA
9	25	000915	E0 A4 95	क ्	DEVANAGARI LETTER KA
10	28	00093E	E0 A4 BE	ा	DEVANAGARI VOWEL SIGN AA
11	31	000930	E0 A4 B0	र	DEVANAGARI LETTER RA
12	34	000020	20		SPACE
13	35	00092A	E0 A4 AA	प	DEVANAGARI LETTER PA
14	38	000941	E0 A5 81	ु	DEVANAGARI VOWEL SIGN U
15	41	000938	E0 A4 B8	स ु	DEVANAGARI LETTER SA
16	44	00094D	E0 A5 8D	्	DEVANAGARI SIGN VIRAMA
17	47	000924	E0 A4 A4	त ्	DEVANAGARI LETTER TA
18	50	000915	E0 A4 95	क	DEVANAGARI LETTER KA
19	53	00093E	E0 A4 BE	ा	DEVANAGARI VOWEL SIGN AA
20	56	000932	E0 A4 B2	ल	DEVANAGARI LETTER LA
21	59	00092F	E0 A4 AF	य	DEVANAGARI LETTER YA

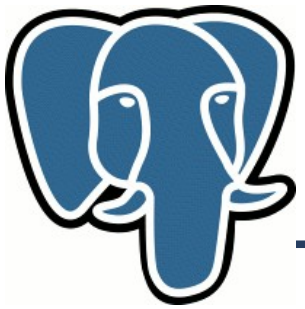


Devanagari script support

8.4 knows Virama signs

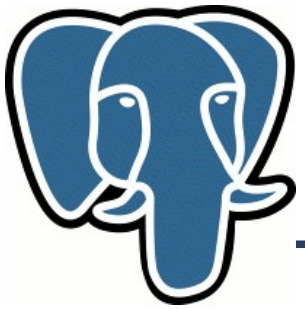
```
=# select * from ts_parse('default',
                                'मदन पुरस्कार पुस्तकालय');
 tokid |  token
-------+---------
     2 | मदन
    12 |
     2 | पुरस्कार
    12 |
     2 | पुस्तकालय
(5 rows)
```

Thanks to Dibyendra Hyoju and Bal Krishna Bal for testing and valuable discussion



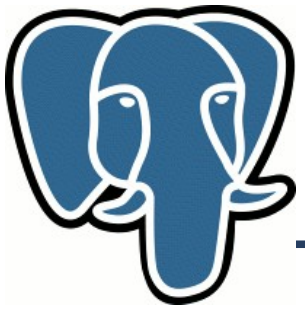
Devanagari script support

- TODO
  - Port stemmer for nepali to snowball
  - Improve Hunspell support (recognize more flags in affix file)



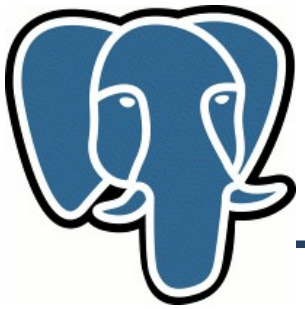
Synonym dictionary with prefix search support (8.5)

```
cat $SHAREDIR/tsearch_data/synonym_sample.syn
postgres        pgsql
postgresql      pgsql
postgre pgsql
gogle   googl
indices index*

=# create text search dictionary syn
( template=synonym,synonyms='synonym_sample');
=# select ts_lexize('syn','indices');
 ts_lexize
-----------
 {index}
```



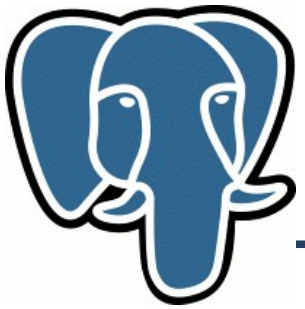
Synonym dictionary with prefix search support (8.5)

```
=# create text search configuration tst ( copy=simple);
=# alter text search configuration tst alter mapping
     for asciiword with syn;

=# select to_tsquery('tst','indices');
 to_tsquery
------------
 'index':*
=# select 'indexes are very useful'::tsvector @@
              to_tsquery('tst','indices');
 ?column?
----------
 t
```



dict_xsyn improvement

- How to search for 'William' and any synonyms 'Will', 'Bill', 'Billy' ? We can:
  - Index only synonyms
  - Index synonyms and original name
  - Index only original name - replace all synonyms. Index size is minimal, but s*earch for specific name is impossible*.



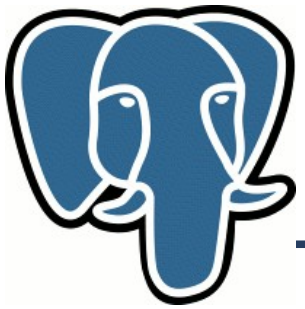
dict_xsyn improvement

- Old version of dict_xsyn can return only list of synonyms. It's possible to prepare synonym file to support other options:

```
William Will Bill Billy
Will William Bill Billy
Bill William Will Billy
Billy William Will Bill
```

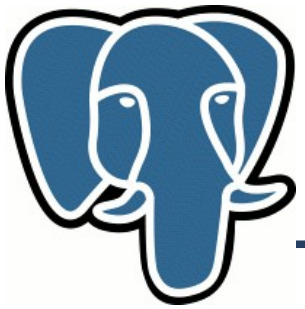
- New dict_xsyn (Sergey Karpov) allows better control:

```
CREATE TEXT SEARCH DICTIONARY xsyn
(RULES='xsyn_sample', KEEPORIG=false|true,
mode='SIMPLE|SYMMETRIC|MAP');
```



dict_xsyn improvement

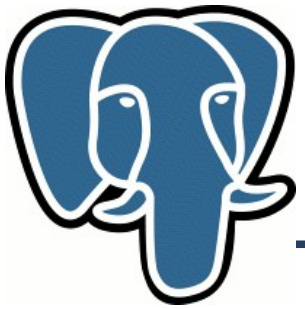
- Mode SIMPLE - accepts the original word and returns all synonyms as OR-ed list. This is default mode.
- Mode SYMMETRIC - accepts the original word **or any** of its synonyms, and return all others as OR-ed list.
- Mode MAP - accepts any synonym and returns the original word.



dict_xsyn improvement

EXAMPLES:

```
=# ALTER TEXT SEARCH DICTIONARY xsyn (RULES='xsyn_sample',
KEEPORIG=false, mode='SYMMETRIC');

=# select ts_lexize('xsyn','Will') as Will,
          ts_lexize('xsyn','Bill') as Bill,
          ts_lexize('xsyn','Billy') as Billy;

         will          |          bill          |         billy
-----------------------+------------------------+-----------------------
 {william,bill,billy} | {william,will,billy} | {william,will,bill}

Mode='MAP'

   will    |   bill    |   billy
-----------+-----------+-----------
 {william} | {william} | {william}
```



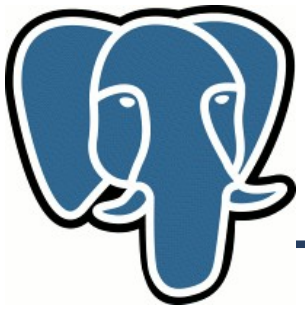
ts_stat() performance !

- ts_stat() function gathers words statistics from tsvectors – now uses binary tree instead of sorted arrays (probably, better to use rbtree to defense against skewed data)

Dataset with geonames, total 5,793,013 rows with 2,404,197 unique names:

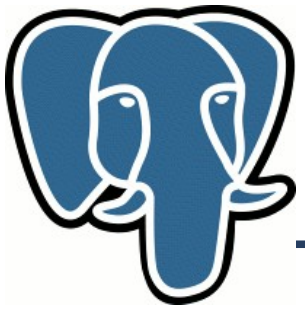
```
=# select * into ts_stat2
     from ts_stat('select fts from spots');
```

8.3: 66405972.737 ms
CVS HEAD: 25506.736 ms 2600x faster !



Fast approximated statistics

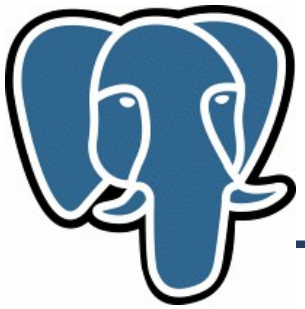
- Gevel extension — GiST/GIN indexes explorer (http://www.sai.msu.su/~megera/wiki/Gevel)
- **Fast** — uses only GIN index (no table access)
- **Approximated** — no table access, which contains visibility information, approx. for long posting lists
- For mostly **read-only** data error is small



Fast approximated statistics

- Top-5 most frequent words (463,873 docs)

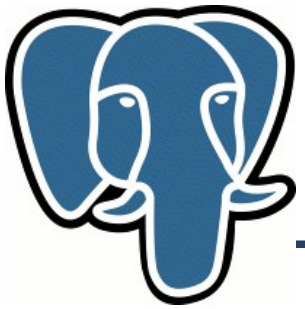
```
=# SELECT * FROM gin_stat('gin_idx') as t(word text, ndoc int)
order by ndoc desc limit 5;

  word  |  ndoc
--------+--------
 page   | 340858
 figur  | 240366
 use    | 148022
 model  | 134442
 result | 129010
(5 rows)
Time: 520.714 ms
```



Fast approximated statistics

- gin_stat() vs ts_stat()

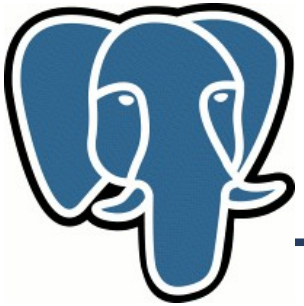
```
=# select * into stat from ts_stat('select fts from papers') order by ndoc
desc, nentry desc,word;

...wait.... 68704,182 ms

=# SELECT a.word, b.ndoc as exact, a.estimation as estimation,
round ( (a.estimation-b.ndoc)*100.0/a.estimation,2)||'%' as error
FROM (SELECT * FROM gin_stat('gin_x_idx') as t(word text, estimation int)
order by estimation desc limit 5 ) as a, stat b
WHERE a.word = b.word;

  word  | exact  | estimation | error
--------+--------+------------+-------
 page   | 340430 |     340858 | 0.13%
 figur  | 240104 |     240366 | 0.11%
 use    | 147132 |     148022 | 0.60%
 model  | 133444 |     134442 | 0.74%
 result | 128977 |     129010 | 0.03%
(5 rows)
Time: 550.562 ms
```



GIN improvements

- GIN fast update (8.4)
- GIN partial match support (8.4)
- GIN multicolumn index (8.4)
- contrib/btree_gin (8.4) – provides GIN operator classes, that implement B-tree for all data types. Useful to use with GIN multicolumn feature:

```
CREATE index fts_idx ON papers USING
 gin(timestamp, fts_tsvector);
```

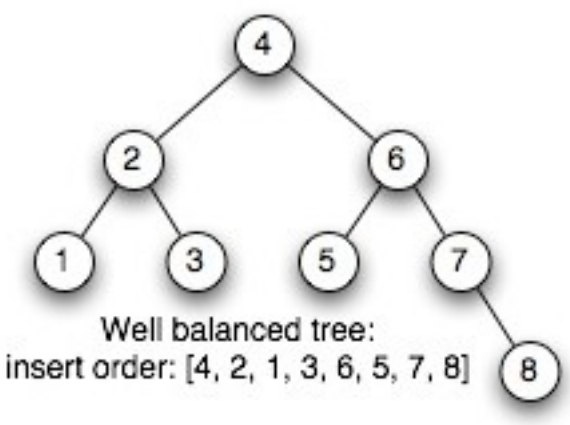


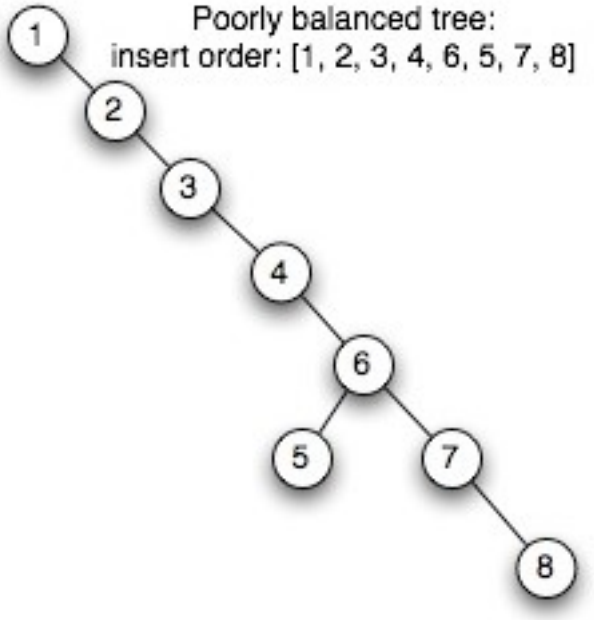
Talk roadmap

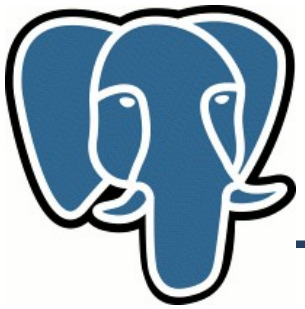
- Full-text search introduction
- Main topics
  - Phrase Search
  - Dictionaries API
- New features (already in 8.4)
- Future features
- Tips and Tricks

Future features

Red-Black tree experiment to replace binary tree in GIN – better defense against skewed data.







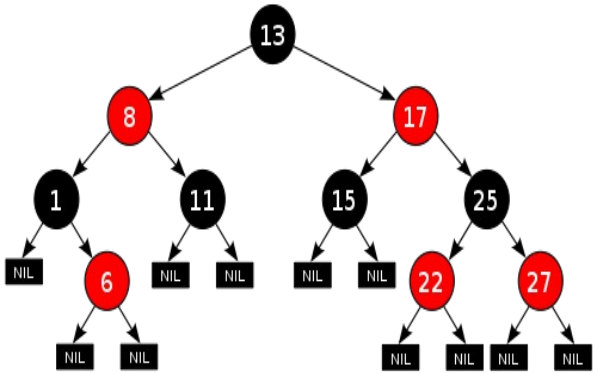
Future features

- Red-Black tree experiment to replace binary tree in GIN – better defense against skewed data – motivational example by Sergey Burladyan
  http://archives.postgresql.org/pgsql-performance/2009-03/msg00340.php

```
create table a (i1 int, i2 int, i3 int, i4 int, i5 int, i6 int);
insert into a select n, n, n, n, n, n from generate_series(1, 100000) as n;
create index arr_gin on a using gin ( (array[i1, i2, i3, i4, i5, i6]) );

truncate a;
drop index arr_gin ;
create index arr_gin on a using gin ( (array[i1, i2, i3, i4, i5, i6]) );
insert into a select n, n, n, n, n, n from generate_series(1, 100000) as n;
```

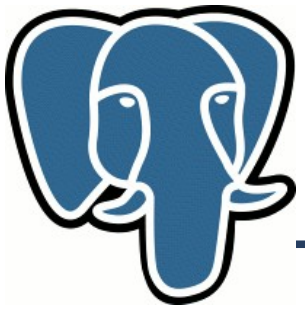
Red-Black Tree

- 8.3.5 – binary tree
- 8.4beta1 - binary tree + limit
- 8.4beta1+Red-Black tree



(self-balancing binary search tree,
the longest path from any node to a leaf
is no more than twice the shortest path)

	8.3.5	8.4beta1	8.4beta1+rbtree
index (bulk):	123276.419	2686.435	2075.634
index+insert:	3415.676	2900.268	2708.512



Red-Black Tree

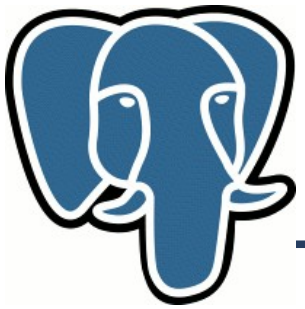
```
select array_to_string(ARRAY(select '' || c || '.' || b from
generate_series(1,50) b), ' ')::tsvector AS i INTO a
FROM generate_series(1,100000) c;
create index arr_gin on a using gin (i);
```

```
drop table a;
create table a ( i tsvector);
```

'1.1' '1.10' '1.11' '1.12' '1.13' '1.14' ....
'2.1' '2.10' '2.11' '2.12' '2.13' '2.14' ....

```
create index arr_gin on a using gin (i);
insert into a select array_to_string(ARRAY(select '' || c || '.' || b from
generate_series(1,50) b), ' ')::tsvector AS i
FROM generate_series(1,100000) c;
```

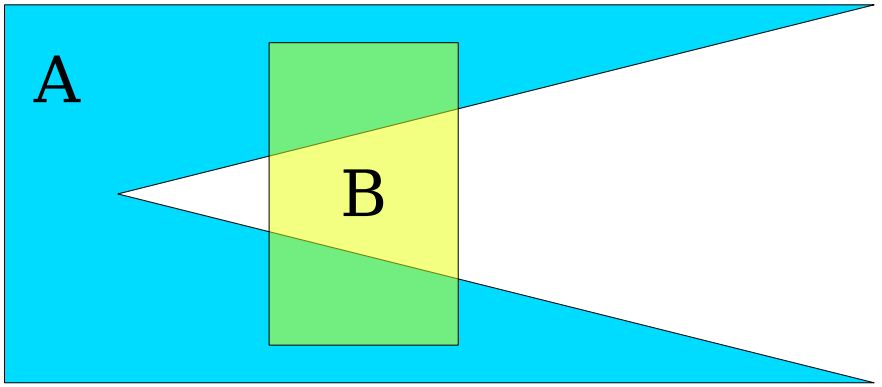
	8.3.5	8.4beta1	8.4beta1+rbtree
index (bulk):	inf (>1night)	228564.291	152569.763
index+insert:	410300.855	314332.507	251015.830
Epaper archive:		81714.308	86312.517



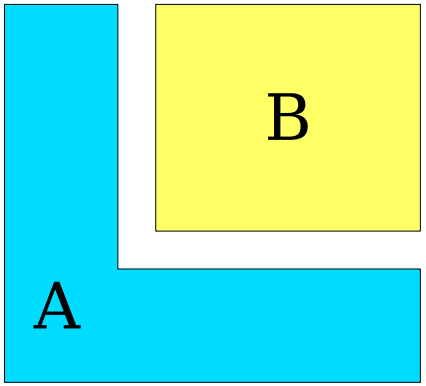
Downloads (CVS HEAD)

- Phrase search
  - http://www.sigaev.ru/misc/phrase_search-0.7.gz
- Filter dictionary support
  - http://www.sigaev.ru/misc/filter_dict-0.2.gz
  - http://www.sigaev.ru/misc/unaccent-0.2.tar.gz
- Synonym dictionary with prefix search
  - http://www.sigaev.ru/misc/synonym_prefix.gz
- Red-Black tree
  - http://www.sigaev.ru/misc/rbtree-0.2.gz

Polygons

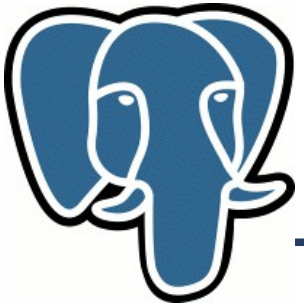


A @> B = TRUE



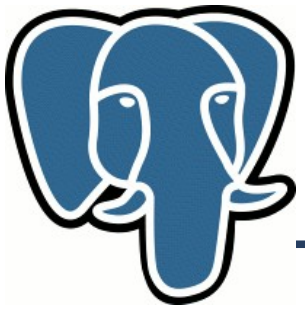
A && B = TRUE

http://www.sigaev.ru/misc/polygon-0.1.gz



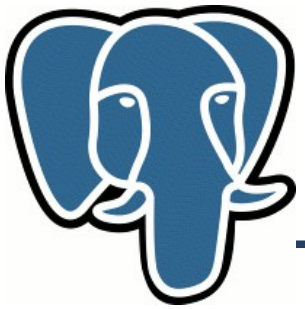
Talk roadmap

- Full-text search introduction
- Main topics
  - Phrase Search
  - Dictionaries API
- New features (already in 8.4)
- Future features
- **Tips and Tricks**



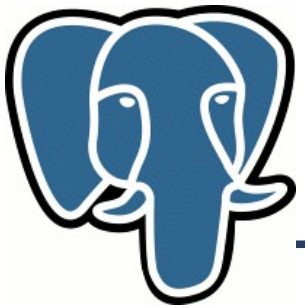
Full-text search tips

- Aggregate for tsvector
- Stable to_tsquery
- Find documents with specific token type
- Getting words from tsvector
- Confuse with text search



Aggregate for tsvector

```
CREATE AGGREGATE tsvector_sum(tsvector) (
    SFUNC = tsvector_concat,
    STYPE = tsvector,
    INITCOND = ''
);
=# SELECT tsvector_sum( t.fts)  FROM  ( select ('1 2 ' ||
generate_series(3,10,1))::tsvector AS fts ) AS t;
            tsvector_sum
-----------------------------------------
 '1' '2' '3' '4' '5' '6' '7' '8' '9' '10'
```



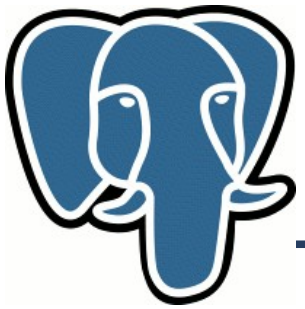
Stable to_tsquery

Result of to_tsquery() can't be used as a cache key, since ts_query() does preserve an order, which isn't good for cacheing.

Little function helps:

```
CREATE OR REPLACE FUNCTION stable_ts_query(tsquery)
RETURNS tsquery AS
$$
   SELECT ts_rewrite( $1 , 'dummy_word', 'dummy_word');
$$
LANGUAGE SQL RETURNS NULL ON NULL INPUT IMMUTABLE;
```

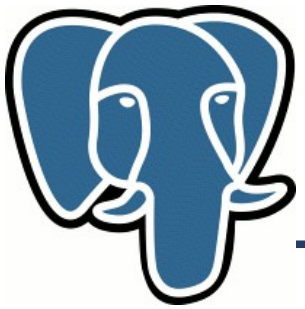
Note: Remember about text search configuraton to have really good cache key !



Find documents with specific token type

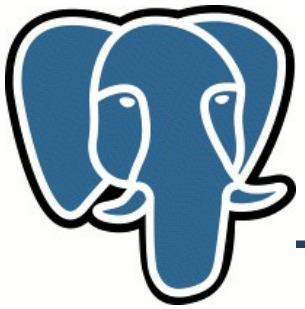
How to find documents, which contain emails ?

```
CREATE OR REPLACE FUNCTION document_token_types(text)
RETURNS _text AS
$$

SELECT ARRAY (
    SELECT
        DISTINCT alias
    FROM
        ts_token_type('default') AS tt,
        ts_parse('default', $1) AS tp
    WHERE
        tt.tokid = tp.tokid
    );
$$ LANGUAGE SQL immutable;
```



Find documents with specific token type

```
=# SELECT document_token_types(title) FROM papers
        LIMIT 10;
                      document_token_types
-----------------------------------------------------------------
 {asciihword,asciiword,blank,hword_asciipart}
 {asciiword,blank}
 {asciiword,blank}
 {asciiword,blank}
 {asciiword,blank}
 {asciiword,blank,float,host}
 {asciiword,blank}
 {asciihword,asciiword,blank,hword_asciipart,int,numword,uint}
 {asciiword,blank}
 {asciiword,blank}
(10 rows)

CREATE INDEX fts_types_idx ON papers USING
                        gin( document_token_types (title) );
```



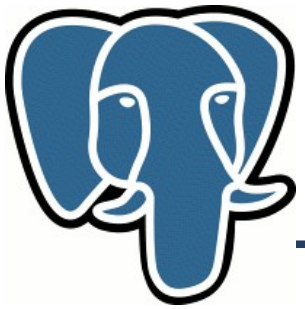
Find documents with specific token type

How to find documents, which contain emails ?

```
SELECT comment FROM papers
WHERE document_token_types(title) && '{email}';
```

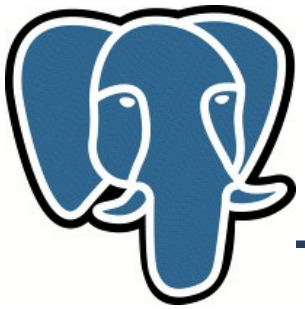
The list of available token types:

```
SELECT * FROM ts_token_type('default');
```



Getting words from tsvector

```
CREATE OR REPLACE FUNCTION ts_stat(tsvector, OUT word text,
OUT ndoc integer, OUT nentry integer)
RETURNS SETOF record AS $$
SELECT ts_stat('SELECT ' || quote_literal( $1::text )
                                 || '::tsvector');
$$ LANGUAGE SQL RETURNS NULL ON NULL INPUT IMMUTABLE;

SELECT id, (ts_stat(fts)).* FROM apod WHERE id=1;
 id |    word     | ndoc | nentry
----+-------------+------+--------
  1 | 1           |    1 |      1
  1 | 2           |    1 |      2
  1 | io          |    1 |      2
  1 | may         |    1 |      1
  1 | new         |    1 |      1
  1 | red         |    1 |      1
  1 | two         |    1 |      1
```



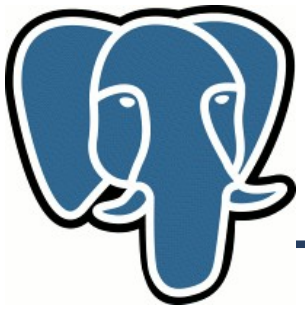
Confuse with text search

One expected **true** here, but result is disappointing **false**

```
=# select to_tsquery('ob_1','inferences') @@
                to_tsvector('ob_1','inference');
 ?column?
----------
 f
```

Use ts_debug() to understand the problem

```
'inferences':
{french_ispell,french_stem} | french_stem   | {inferent}

'inference':
{french_ispell,french_stem} | french_ispell | {inference}
```

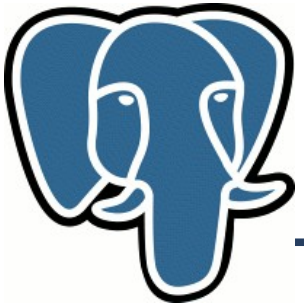


Confuse with text search

- Use synonym dictionary as a first dictionary {synonym,french_ispell,french_stem}

  with rule '`inferences inference`'

  - Don't forget to reindex !
- Use ts_rewrite()
  - Don't need to reindex



- Our work was supported by
  - Russian Foundation for Basic Research
  - EnterpriseDB
  - jfg://networks

THANKS !