Full-Text Search in PostgreSQL

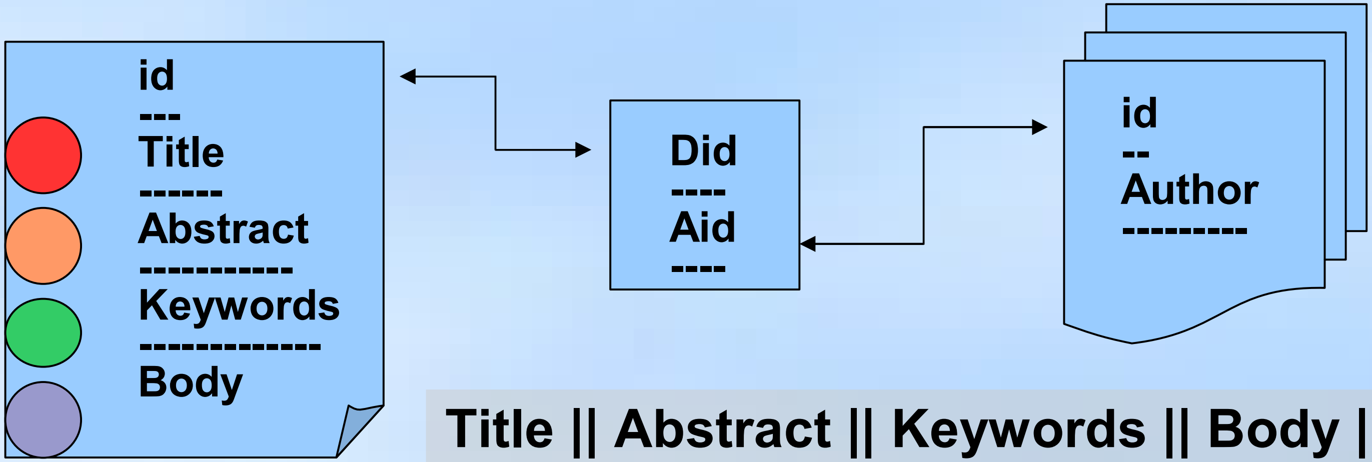
Oleg Bartunov
Moscow University
PostgreSQL Global Development Group

FTS in Database

- **Full-text search**
  - Find documents, which *satisfy* query
  - return results in some order (opt.)
- **Requirements to FTS**
  - **Full integration with PostgreSQL**
    - transaction support
    - concurrency and recovery
    - online index
  - **Linguistic support**
  - **Flexibility**
  - **Scalability**

What is a Document ?

- Arbitrary textual attribute
- Combination of textual attributes
- Should have unique id
- Could be fully virtual
- It's a textual result of any SQL command



Title || Abstract || Keywords || Body || Author

Text Search Operators

- Traditional FTS operators for textual attributes **~, ~*, LIKE, ILIKE**

Problems

– No linguistic support, no stop-words

– No ranking

– Slow, no index support. Documents should be scanned every time.

Solution

– Preprocess document in advance

– Add index support

FTS in PostgreSQL

```
=# select  'a fat cat sat on a mat and ate a fat rat'::tsvector
                                    @@
                          'cat & rat':: tsquery;
```

- **tsvector** – storage for document, optimized for search
  - sorted array of lexemes
  - positional information
  - weights information
- **tsquery** – textual data type for query
  - Boolean operators - & | ! ()
- **FTS operator**

  tsvector @@ tsquery

FTS in PostgreSQL

- FTS is consists of
  - set of rules, which define how document and query should be transformed to their FTS representations – tsvector, tsquery.
  - set of functions to obtain tsvector, tsquery from textual data types
  - FTS operators and indexes
  - ranking functions, headline
- OpenFTS - openfts.sourceforge.net
  - constructs tsvector, tsquery by itself
  - use FTS operator and indexes

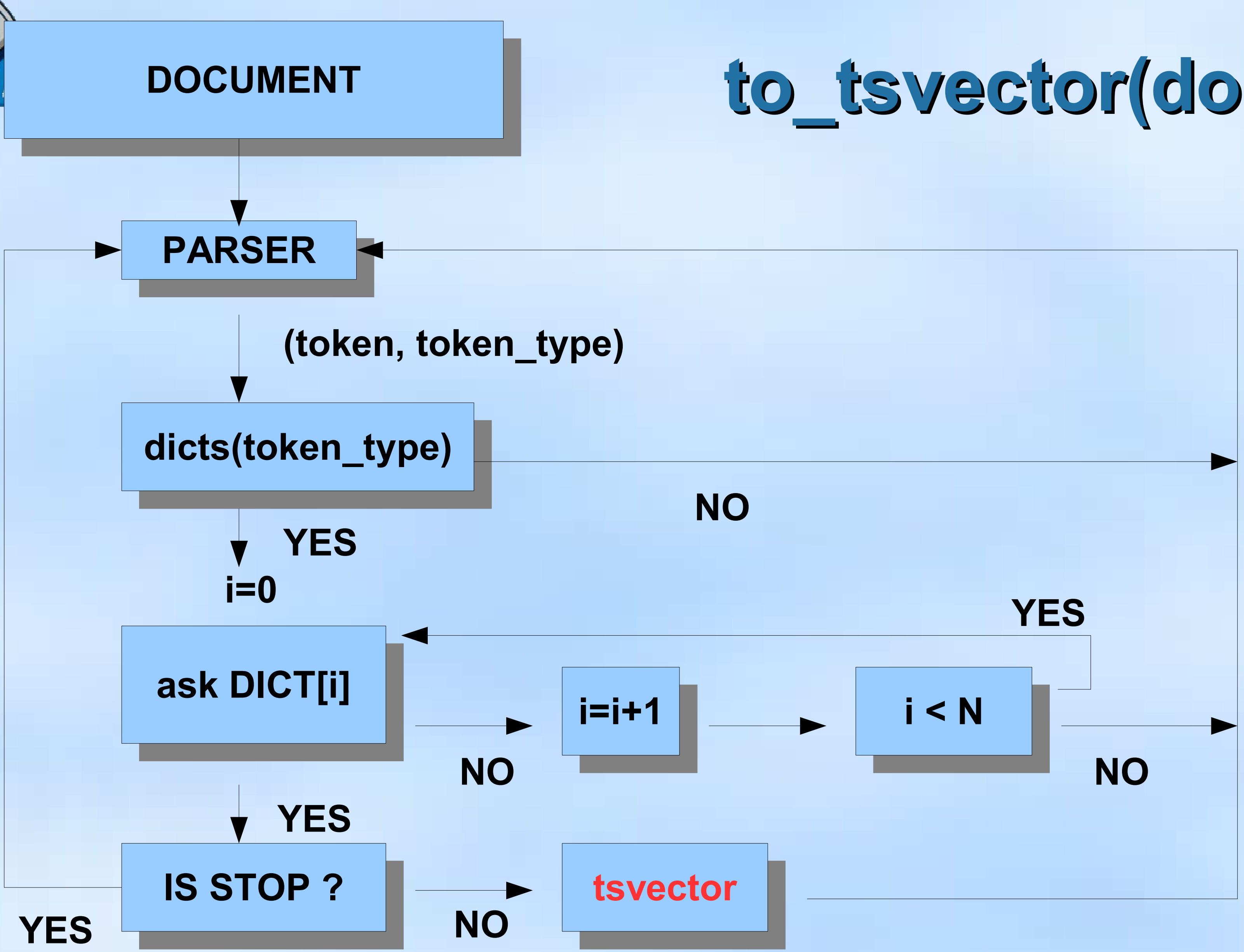
FTS features

- Full integration with PostgreSQL
- 27 built-in configurations for 10 languages
- Support of user-defined FTS configurations
- Pluggable dictionaries (ispell, snowball, thesaurus), parsers
- Multibyte support (UTF-8)
- Relevance ranking
- Two types of indexes – GiST and GiN with concurrency and recovery support
- Rich query language with query rewriting support

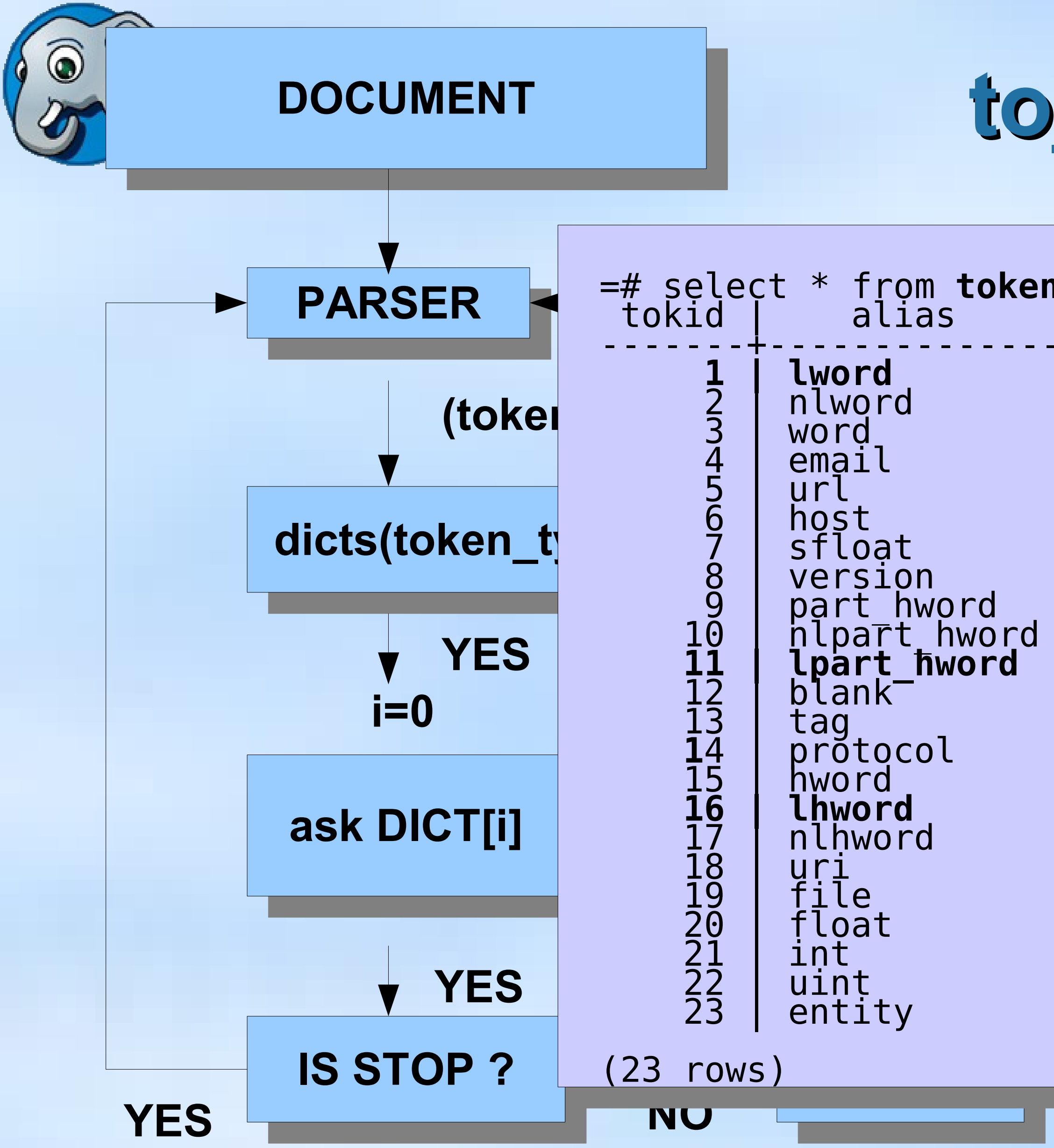
Complete FTS reference

- Data types
  - tsvector, tsquery
- FTS operators
  - @@, @@@
- Basic functions
  - to_tsvector, setweight, to_tsquery, plainto_tsquery, rewrite, tsearch
- Additional functions
  - rank_cd, rank, headline
- Additional operators
  - @>, <@
- Debug functions
  - lexize, ts_debug, parse, token_type, numnode, querytree, stat

to_tsvector(doc)



to_tsvector(doc)



DOCUMENT

PARSER

(toke

dicts(token_t

YES

i=0

ask DICT[i]

YES

IS STOP ?

YES

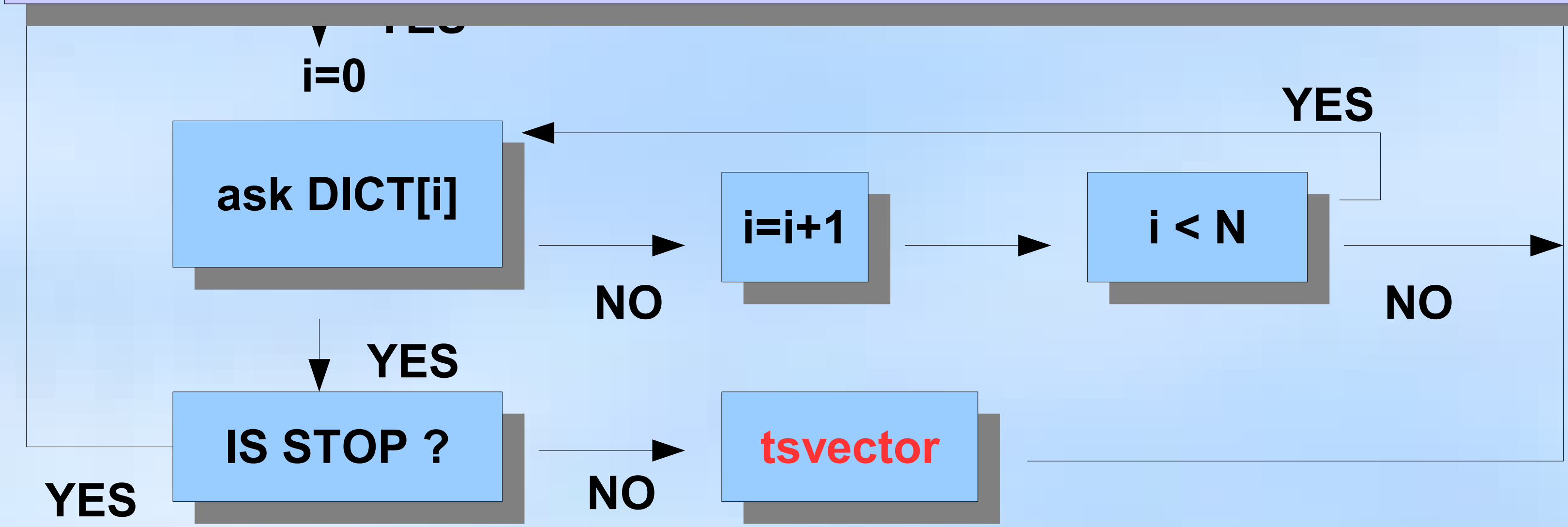
NO

```
=# select * from token_type('default');
 tokid |    alias     |               description
-------+--------------+-----------------------------------
     1 | lword        | Latin word
     2 | nlword       | Non-latin word
     3 | word         | Word
     4 | email        | Email
     5 | url          | URL
     6 | host         | Host
     7 | sfloat       | Scientific notation
     8 | version      | VERSION
     9 | part_hword   | Part of hyphenated word
    10 | nlpart_hword | Non-latin part of hyphenated word
    11 | lpart_hword  | Latin part of hyphenated word
    12 | blank        | Space symbols
    13 | tag          | HTML Tag
    14 | protocol     | Protocol head
    15 | hword        | Hyphenated word
    16 | lhword       | Latin hyphenated word
    17 | nlhword      | Non-latin hyphenated word
    18 | uri          | URI
    19 | file         | File or path name
    20 | float        | Decimal notation
    21 | int          | Signed integer
    22 | uint         | Unsigned integer
    23 | entity       | HTML Entity

(23 rows)
```

to_tsvector(doc)

```
    Token        |                  Dictionaries
-----------------+-----------------------------------------------------------
 file            | pg_catalog.simple
 host            | pg_catalog.simple
 hword           | pg_catalog.simple
 int             | pg_catalog.simple
 lhword          | public.pg_dict,public.en_ispell,pg_catalog.en_stem
 lpart_hword     | public.pg_dict,public.en_ispell,pg_catalog.en_stem
 lword           | public.pg_dict,public.en_ispell,pg_catalog.en_stem
 nlhword         | pg_catalog.simple
 nlpart_hword    | pg_catalog.simple
```

```
lexize('en_stem','stars')
-------------------------
         {star}
```



Dictionaries

- **Dictionary** – is a program, which accepts token and returns
  - an array of lexemes, if it is known and not a stop-word
  - void array, if it is a stop-word
  - NULL, if it's unknown
- API for developing specialized dictionaries
- Built-in dictionary-templates :
  - ispell (works with ispell, myspell, hunspell dicts)
  - snowball stemmer
  - synonym, thesaurus
  - simple

Dictionaries

- Dictionary for integers

```
CREATE TEXT SEARCH DICTIONARY intdict
        LEXIZE  'dlexize_intdict' INIT  'dinit_intdict'
        OPTION  'MAXLEN=6,REJECTLONG=false'
;
select lexize('intdict', 11234567890);
  lexize
----------
 {112345}
```

Dictionaries

- Dictionary for roman numerals

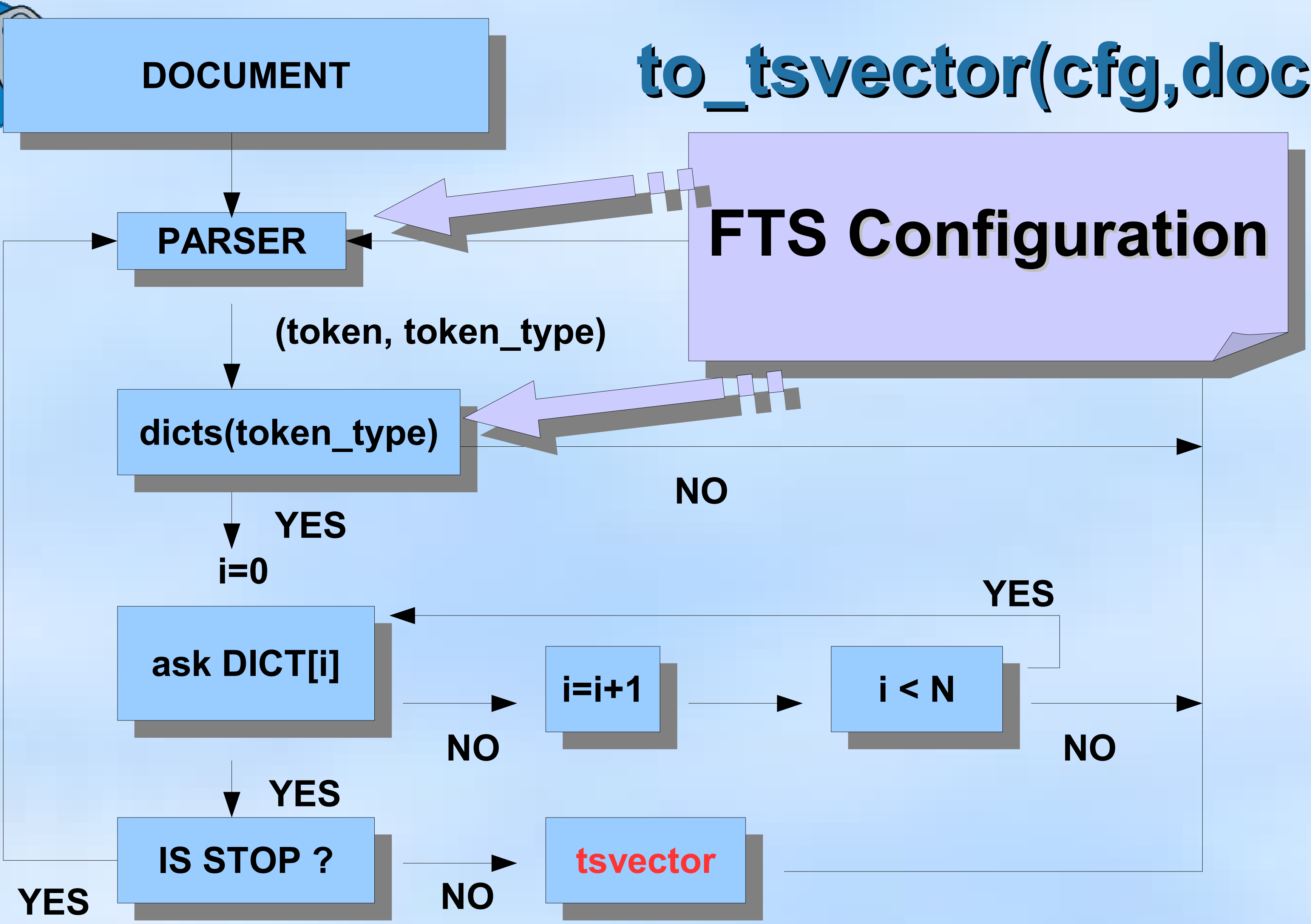
```
=# select lexize('roman', 'XIX');
 lexize
--------
 {19}

=# select  to_tsvector('roman', 'postgresql was born in XIX-century') @@
    plainto_tsquery('roman','19 century');
 ?column?
----------
 t
```

Dictionaries

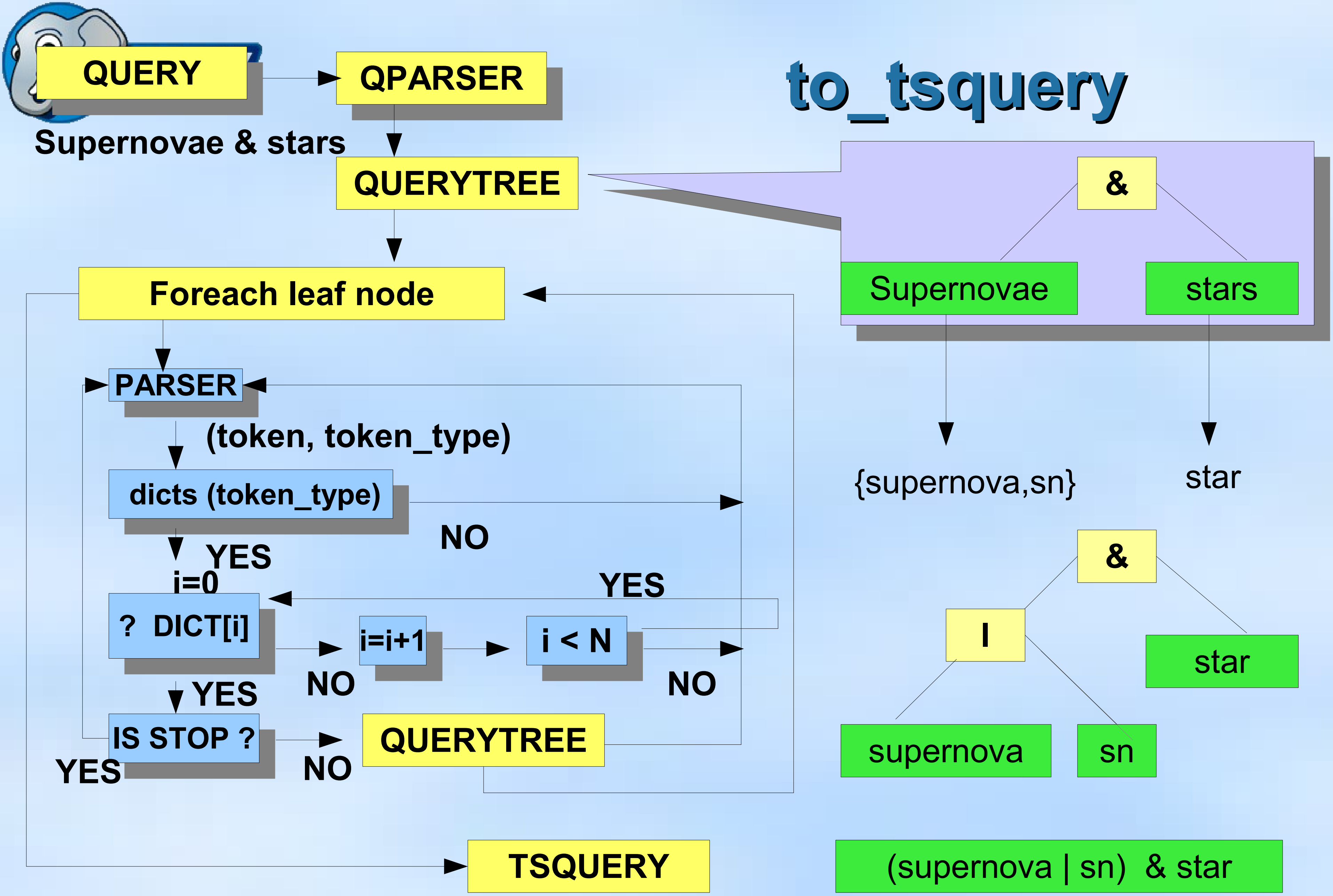
- Dictionary with regexp support (pcre library)

```
# Messier objects
(M|Messier)(\s|-)?((\d){1,3}) M$3
# catalogs
(NGC|Abell|MKN|IC|H[DHR]|UGC|SAO|MWC)(\s|-)?((\d){1,6}[ABC]?) $1$3
(PSR|PKS)(\s|-)?([JB]?)(\d\d\d\d)\s?([+-]\d\d)\d? $1$4$5
# Surveys
OGLE(\s|-)?((I){1,3}) ogle
2MASS twomass
# Spectral lines
H(\s|-)?(alpha|beta|gamma) h$2
(Fe|Mg|Si|He|Ni)(\s|-)?((\d)|([IXV])+) $1$3
# GRBs
gamma\s?ray\s?burst(s?) GRB
GRB\s?(\d\d\d\d\d\d)([abcd]?) GRB$1$2
```

to_tsvector(cfg,doc)



to_tsquery



QUERY

Supernovae & stars

QPARSER

QUERYTREE

Foreach leaf node

PARSER

(token, token_type)

dicts (token_type)

NO

YES

i=0

? DICT[i]

YES

i=i+1

i < N

NO

NO

YES

IS STOP ?

YES

NO

QUERYTREE

TSQUERY

&

Supernovae

stars

{supernova,sn}

star

&

|

star

supernova

sn

(supernova | sn) & star

to_tsquery, plainto_tsquery

- to_tsquery expects *preparsed* text
  - tokens with boolean operators between - & (AND), | (OR), ! (NOT) and parentheses
  - tokens can have weight labels

    'fat:ab & rats & ! (cats | mice)'
- plainto_tsquery accepts *plain text*
- Tip: quote text in to_tsquery

```
select to_tsquery('''supernovae stars'':ab & !crab');
---------------
'sn':AB & !'crab'
```

Indexes

- Indexes speedup full-text operators
  - FTS should works without indexes !
- Two types of indexes
  - GiST index
    - fast update
    - not well scaled with #words, #documents
    - supports **fillfactor** parameter

```
create index gist_idx on apod using gist(fts)
                         with (fillfactor=50);
```

  - GiN index
    - slow update
    - good scalability
- Both indexes support concurrency and recovery

GiST index - Signatures

- Each word hashed to the bit position – word signature

w1 -> S1: 01000000   Document: w1 w2 w3

w2 -> S2: 00010000

w3 -> S3: 10000000

- Document signature is a superposition of word signatures

S: 11010000   S1 || S2 || S3 – bit-wise OR

- Query signature – the same way
- Bloom filter

Q1: 00000001 – exact not

Q2: 01010000 - may be contained in the document, **false drop**

- Signature is a **lossy** representation of a document
  - + fixed length, compact, + fast bit operations
  - - lossy (false drops), - saturation with #words grows

GiST Index

Demo collections – latin proverbs

```
 id |        proverb
----+------------------------
  1 | Ars longa, vita brevis
  2 | Ars vitae
  3 | Jus vitae ac necis
  4 | Jus generis humani
  5 | Vita nostra brevis
```

GiST Index

- Demo collections – latin proverbs
  - each word represented as a fixed-length bitmap

```
   word     | signature
----------+------------
 ac         | 00000011
 ars        | 11000000
 brevis     | 00001010
 generis    | 01000100
 humani     | 00110000
 jus        | 00010001
 longa      | 00100100
 necis      | 01001000
 nostra     | 10000001
 vita       | 01000001
 vitae      | 00011000
```

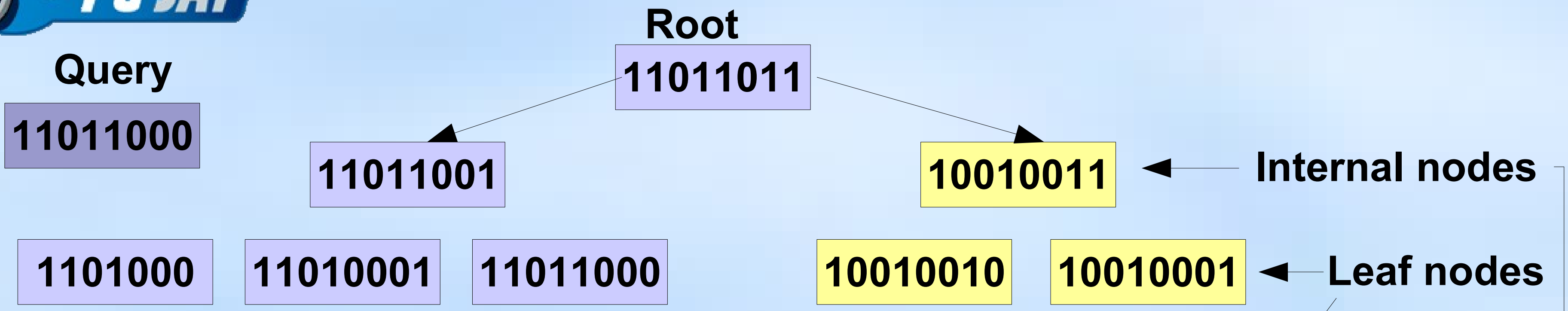
Document is a bitwise-OR of all signatures

```
ars vitae => 11000000
             00011000
             ========
             11011000
```

```
 id |         proverb          | signature
----+--------------------------+----------
  1 | Ars longa, vita brevis    | 11101111
  2 | Ars vitae                 | 11011000
  3 | Jus vitae ac necis        | 01011011
  4 | Jus generis humani        | 01110101
  5 | Vita nostra brevis        | 11001011
```

false hit

GiST index - RD-Tree



```
arxiv=# select * from gist_print('gist_idx_90') as
        t(level int,valid bool, fts gtsvector) where level =4;
 level | valid |               fts
------+------ +--------------------------------
     4 | t      | 130 true bits, 1886 false bits
     4 | t      | 95 unique words
     4 | t      | 33 unique words
....................................
     4 | t      | 61 unique words
(417366 rows)
```

contrib module **Gevel**

```
arxiv=# select * from gist_print('gist_idx_90') as
        t(level int, valid bool, fts gtsvector) where level =3;

 level | valid |               fts
------+-------+--------------------------------
     3 | t       | 852 true bits, 1164 false bits
     3 | t       | 861 true bits, 1155 false bits
     3 | t       | 858 true bits, 1158 false bits
...................................................................
     3 | t       | 773 true bits, 1243 false bits
(17496 rows)
```

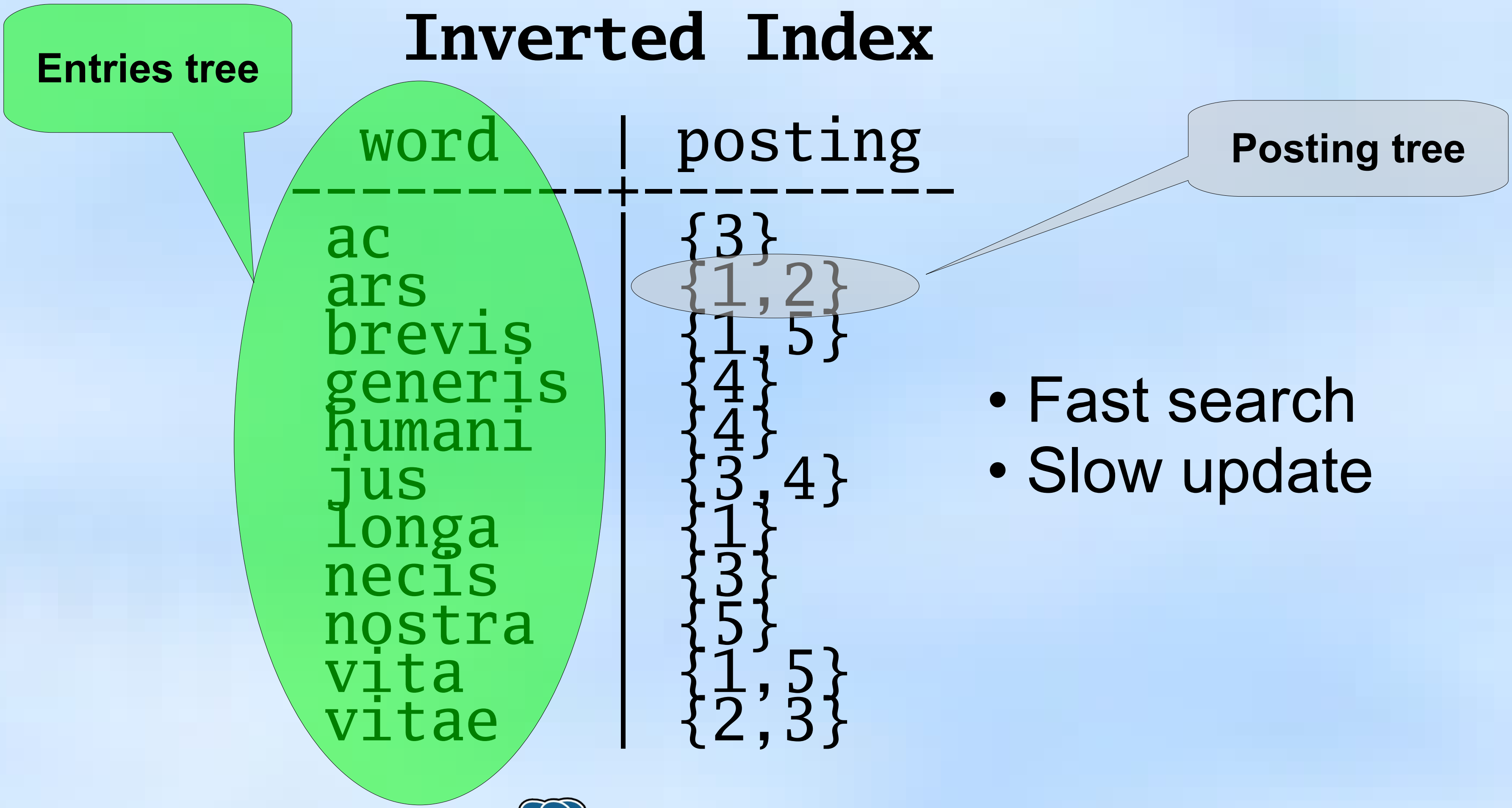
GIN Index

Demo collections – latin proverbs

```
 id |        proverb
----+-----------------------
  1 | Ars longa, vita brevis
  2 | Ars vitae
  3 | Jus vitae ac necis
  4 | Jus generis humani
  5 | Vita nostra brevis
```

GIN Index

Demo collections – latin proverbs



```
         Inverted Index
   word    |  posting
-----------+-----------
  ac       | {3}
  ars      | {1,2}
  brevis   | {1,5}
  generis  | {4}
  humani   | {4}
  jus      | {3,4}
  longa    | {1}
  necis    | {3}
  nostra   | {5}
  vita     | {1,5}
  vitae    | {2,3}
```

- Fast search
- Slow update

GiN or GiST ?

Direct comparison of performance on abstracts from e-print archives

Total number of abstracts - 405690.
Desktop PC, P4 2.4Ghz, 2Gb RAM, Linux 2.6.19.1, Slackware,PostgreSQL 8.2.4.

postgresql.conf:

```
shared_buffers = 256MB
work_mem = 8MB
maintenance_work_mem = 64MB
checkpoint_segments = 9
effective_cache_size = 256MB
```

```
arxiv=# select pg_relation_size('papers');
 pg_relation_size
------------------
       1054081024

arxiv=# select count(*) from wordstat;
 count
--------
 459841
```

GiN or GiST ?

query 'gamma & ray & burst & !supernovae' – 2764 hits

```
index          creation(ms)    size (b)        count(*)        rank query
-------------------------------------------------------------------------
GiN             532310.368      305864704       38.739          130.488
GIST90          176267.543      145989632       111.891         188.992
GIST100         189321.561      130465792       120.730         215.153
GIST50          164669.614      279306240       122.101         200.963

Updating:
index (nlev)      95            1035            10546
-----------------------------------------------------------
GIN             3343.881        36337.733       217577.424
GIST90  (4)     280.072         1835.485        29597.235
GIST100 (4)     232.674         2460.621        27852.507
GIST50  (5)     238.101         2952.362        33984.443
```

Conclusions:

- creation time - GiN takes 3x time to build than GiST
- size of index - GiN is 2-3 times bigger than GiST
- search time - GiN is 3 times faster than GiST
- update time - GiN is about 10 times slower than GiST

FTS new features

- FTS configuration - schema support
- FTS operator for textual data types
- Correct dump/restore (*)
- SQL interface to FTS configuration
- psql commands to display info about FTS objects
- changes of FTS objects are immediate
- ispell supports ispell, myspell, hunspell dicts
- improved ts_debug
- relative paths for dictionary files ($PGROOT/share)

Simple FTS

- FTS operator supports text data types
  - easy FTS without ranking
  - use other ordering

```
arxiv=# \d papers
          Table "public.papers"
      Column       |   Type    | Modifiers
-------------------+-----------+-----------
 id                | integer   |
 oai_id            | text      |
 datestamp         | date      |
 title             | text      |
 modification_date | date      |
```

```
arxiv=# create index title_idx on papers using gin(title);
```

```
arxiv=# select title from papers p where title @@
  to_tsquery('supernovae & (Ia | Ib)')
  order by modification_date desc limit 5;
```

FTS without tsvector column

- Use functional index (GiST or GiN)
  - – no ranking, use other ordering

```
create index gin_text_idx on test using gin (
( coalesce(to_tsvector(title),'') || coalesce(to_tsvector(body),'') )
);
```

```
apod=# select title from test where
(coalesce(to_tsvector(title),'') || coalesce(to_tsvector(body),'') ) @@
to_tsquery('supernovae') order by sdate desc limit 10;
```

APOD example

- curl -O http://www.sai.msu.su/~megera/postgres/fts/apod.dump.gz
- zcat apod.dump.gz | psql postgres
- psql postgres

```
postgres=# \d apod
       Table "public.apod"
  Column   |   Type    | Modifiers
----------+-----------+-----------
 id        | integer   | not null
 title     | text      |
 body      | text      |
 sdate     | date      |
 keywords | text      |
```

```
postgres=# show tsearch_conf_name;
  tsearch_conf_name
-------------------------
 pg_catalog.russian_utf8
```

APOD example

```
postgres=# \dF+ pg_catalog.russian_utf8
Configuration "pg_catalog.russian_utf8"
Parser name: "pg_catalog.default"
Locale: 'ru_RU.UTF-8' (default)
    Token     |      Dictionaries
--------------+-------------------------
 email        | pg_catalog.simple
 file         | pg_catalog.simple
 float        | pg_catalog.simple
 host         | pg_catalog.simple
 hword        | pg_catalog.ru_stem_utf8
 int          | pg_catalog.simple
 lhword       | pg_catalog.en_stem
 lpart_hword  | pg_catalog.en_stem
 lword        | pg_catalog.en_stem
 nlhword      | pg_catalog.ru_stem_utf8
 nlpart_hword | pg_catalog.ru_stem_utf8
 nlword       | pg_catalog.ru_stem_utf8
 part_hword   | pg_catalog.simple
 sfloat       | pg_catalog.simple
 uint         | pg_catalog.simple
 uri          | pg_catalog.simple
 url          | pg_catalog.simple
 version      | pg_catalog.simple
 word         | pg_catalog.ru_stem_utf8
```

APOD example

```
postgres=# alter table apod add column fts tsvector;
postgres=# update apod  set fts=
                               setweight( coalesce( to_tsvector(title),''),'B')          ||
                               setweight( coalesce( to_tsvector(keywords),''),'A') ||
                               setweight( coalesce( to_tsvector(body),''),'D');
```

NULL || nonNULL => NULL

if NULL then ''

```
postgres=# create index apod_fts_idx on apod using gin(fts);
postgres=# vacuum analyze apod;

postgres=# select title from apod where fts  @@ plainto_tsquery('supernovae stars') limit 5;
                 title
------------------------------------------
 Runaway Star
 Exploring The Universe With IUE 1978-1996
 Tycho Brahe Measures the Sky
 Unusual Spiral Galaxy M66
 COMPTEL Explores The Radioactive Sky
```

APOD example: Search

```
postgres=# select title,rank_cd(fts, q) from apod,
to_tsquery('supernovae & x-ray') q
where fts  @@ q order by rank_cd desc limit 5;
                     title                        | rank_cd
--------------------------------------------------+---------
 Supernova Remnant E0102-72 from Radio to X-Ray   | 1.59087
 An X-ray Hot Supernova in M81                    | 1.47733
 X-ray Hot Supernova Remnant in the SMC           | 1.34823
 Tycho's Supernova Remnant in X-ray               | 1.14318
 Supernova Remnant and Neutron Star               | 1.08116
(5 rows)

Time: 1.965 ms
```

rank_cd uses only local information !

0 < rank/(rank+1) < 1

rank_cd('{0.1, 0.2, 0.4, 1.0 }',fts, q)
D C B A

APOD example: headline

```
postgres=# select headline(body,q,'StartSel=<,StopSel=>,MaxWords=10,MinWords=5'),
rank_cd(fts, q) from apod, to_tsquery('supernovae & x-ray') q where fts  @@
q order by rank_cd desc limit 5;
                               headline                                | rank_cd
-----------------------------------------------------------------------+---------
 <supernova> remnant  E0102-72, however, is giving astronomers a clue | 1.59087
 <supernova> explosion. The picture was taken in  <X>-<rays>          | 1.47733
  <X>-<ray> glow is produced by  multi-million degree                 | 1.34823
  <X>-<rays> emitted by this shockwave made by a telescope            | 1.14318
  <X>-<ray> glow. Pictured is the <supernova>                         | 1.08116
(5 rows)

Time: 39.298 ms
```

Slow, use subselects ! See tips

APOD example

- Different searches with one full-text index
  - title search

```
=# select title,rank_cd(fts, q) from apod,
to_tsquery('supernovae:b & x-ray') q
where fts  @@@ q order by rank_cd desc limit 5;
                     title                      | rank_cd
------------------------------------------------+---------
 Supernova Remnant E0102-72 from Radio to X-Ray | 1.59087
 An X-ray Hot Supernova in M81                  | 1.47733
 X-ray Hot Supernova Remnant in the SMC         | 1.34823
 Tycho's Supernova Remnant in X-ray             | 1.14318
 Supernova Remnant and Neutron Star             | 1.08116
(5 rows)
```

to_tsquery('supernovae:ab') - title and keywords search

FTS tips

- headline() function is slow – use **subselect**

790 times

```
select id,headline(body,q),rank(fts,q) as rank
from apod, to_tsquery('stars') q
where fts @@ q order by rank desc limit 10;

Time: 723.634 ms
```

10 times !

```
select id,headline(body,q),rank from (
  select id,body,q, rank(fts,q) as rank from apod,
  to_tsquery('stars') q
  where fts @@ q order by rank desc limit 10
) as foo;

Time: 21.846 ms
```

```
=#select count(*)from apod where fts @@ to_tsquery('stars');
 count
-------
   790
```

FTS tips

- Fuzzy search with contrib/pg_trgm - trigram statistics

```
=# select show_trgm('supyrnova');
                    show_trgm
-------------------------------------------------
 {"  s"," su",nov,ova,pyr,rno,sup,upy,"va ",yrn}
```

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

=# \d apod_words
  Table "public.apod_words"
 Column |  Type   | Modifiers
--------+---------+-----------
 word   | text    |
 ndoc   | integer |
 nentry | integer |

=# create index trgm_idx on apod_words using gist(word gist_trgm_ops);
=# select word, similarity(word, 'supyrnova') AS sml
from apod_words where word % 'supyrnova' order by sml desc, word;
   word    |   sml
-----------+----------
 supernova | 0.538462
```

collect statistics

To be or not to be ...

Two FTS configurations:
with and without stop-words

To be or not to be ...

```
hamlet=# \dFd+ en_stem
                                        List of fulltext dictionaries
   Schema    |   Name    | Init method  | Lexize method |       Init options         |
-------------+-----------+--------------+---------------+----------------------------+---
 pg_catalog | en_stem | dsnb_en_init | dsnb_lexize   | dicts_data/english.stop | En
```

```
CREATE TEXT SEARCH DICTIONARY en_stem_nostop  OPTION NULL
     LIKE  en_stem;
CREATE TEXT SEARCH CONFIGURATION hamlet LIKE english WITH MAP;
ALTER TEXT SEARCH CONFIGURATION hamlet
   ALTER MAPPING  lhword,lpart_hword,lword WITH en_stem_nostop;

update text set fts=coalesce(to_tsvector('hamlet',txt),'');
hamlet=# select headline('hamlet',txt,q,'StartSel=<,StopSel=>') from text,
      plainto_tsquery('hamlet','to be or not to be') q  where fts @@ q;
                  headline
---------------------------------------------------
 Ham. <To> <be>, <or> <not> <to> <be>, that is the Question:
```

•FTS tips – Query rewriting

- Online rewriting of query
  - Query expansion
    - synonyms (new york => Gottham, Big Apple, NYC ...)
  - Query narrowing (submarine Kursk went down)
    - Kursk => submarine Kursk
- Similar to synonym (thesaurus) dictionary, but doesn't require reindexing

FTS tips – Query rewriting

rewrite (tsquery, tsquery, tsquery)

rewrite (ARRAY[tsquery,tsquery,tsquery]) from aliases

rewrite (tsquery,'select tsquery,tsquery from aliases')

```
create table aliases( t tsquery primary key, s tsquery);

insert into aliases values(to_tsquery('supernovae'),
to_tsquery('supernovae|sn'));

apod=# select rewrite(to_tsquery('supernovae'),
'select * from aliases');
      rewrite
--------------------
 'supernova' | 'sn'
```

FTS tips – Query rewriting

```
apod=# select title, rank_cd(fts,q,1) as rank
from apod, to_tsquery('supernovae') q
where fts @@ q  order by rank desc limit 10;

                     title                      |   rank
------------------------------------------------+----------
 The Mysterious Rings of Supernova 1987A        | 0.669633
 Tycho's Supernova Remnant in X-ray             | 0.598556
 Tycho's Supernova Remnant in X-ray             | 0.598556
 Vela Supernova Remnant in Optical              | 0.591655
 Vela Supernova Remnant in Optical              | 0.591655
 Galactic Supernova Remnant IC 443              | 0.590201
 Vela Supernova Remnant in X-ray                | 0.589028
 Supernova Remnant: Cooking Elements In The LMC | 0.585033
 Cas A Supernova Remnant in X-Rays              | 0.583787
 Supernova Remnant N132D in X-Rays              | 0.579241
```

FTS tips – Query rewriting

```
apod=# select id, title, rank_cd(fts,q,1) as rank
 from apod, rewrite(to_tsquery('supernovae'), 'select * from aliases') q
where fts @@ q  order by rank desc limit 10;
   id    |                  title                  |   rank
---------+-----------------------------------------+----------
 1162701 | The Mysterious Rings of Supernova 1987A |  0.90054
 1162717 | New Shocks For Supernova 1987A          | 0.738432
 1163673 | Echos of Supernova 1987A                | 0.658021
 1163593 | Shocked by Supernova 1987a              | 0.621575
 1163395 | Moving Echoes Around SN 1987A           | 0.614411
 1161721 | Tycho's Supernova Remnant in X-ray      | 0.598556
 1163201 | Tycho's Supernova Remnant in X-ray      | 0.598556
 1163133 | A Supernova Star-Field                  | 0.595041
 1163611 | Vela Supernova Remnant in Optical       | 0.591655
 1161686 | Vela Supernova Remnant in Optical       | 0.591655
```

```
apod=# select  title, rank_cd(fts,q,1) as rank from apod,
      to_tsquery('supernovae') q where fts @@ q and id=1162717;
             title              |   rank
--------------------------------+----------
 New Shocks For Supernova 1987A | 0.533312
```

FTS tips – Partition your data

- Problem:
  - FTS on very big collection of documents
- Solution:
  - Partition data
    - Table inheritance + Constraint Exclusion – current and one or more archive tables
    - GiST index for current table
    - GiN index for archive table(s)

FTS tips – Partition your data

- Create parent class

```
CREATE TABLE papers_class (
       id integer,
       ............
       creation_date date,
       fts tsvector
);
```

- Create *current* and *archive* tables

```
CREATE TABLE papers (
   CHECK (creation_date >= '2007-01-01'::date)
) INHERITS ( papers_class );

CREATE TABLE paper_archive (
   CHECK (creation_date < '2007-01-01'::date)
) INHERITS ( papers_class );
```

FTS tips – Partition your data

- Create GiST index for *current* table

  ```
  CREATE INDEX gist_idx ON paper USING gist(fts);
  ```

  - Not so big
  - Frequently updated
  - GiST is good for updates and fast enough

- Create GIN index for *archive* table

  ```
  CREATE INDEX gin_idx ON paper_archive USING gin(fts);
  ```

  - May be very big
  - Static
  - GIN is very well scaled

- Don't forget to enable constraint exclusion

  ```
  SET constraint_exclusion=on;
  ```

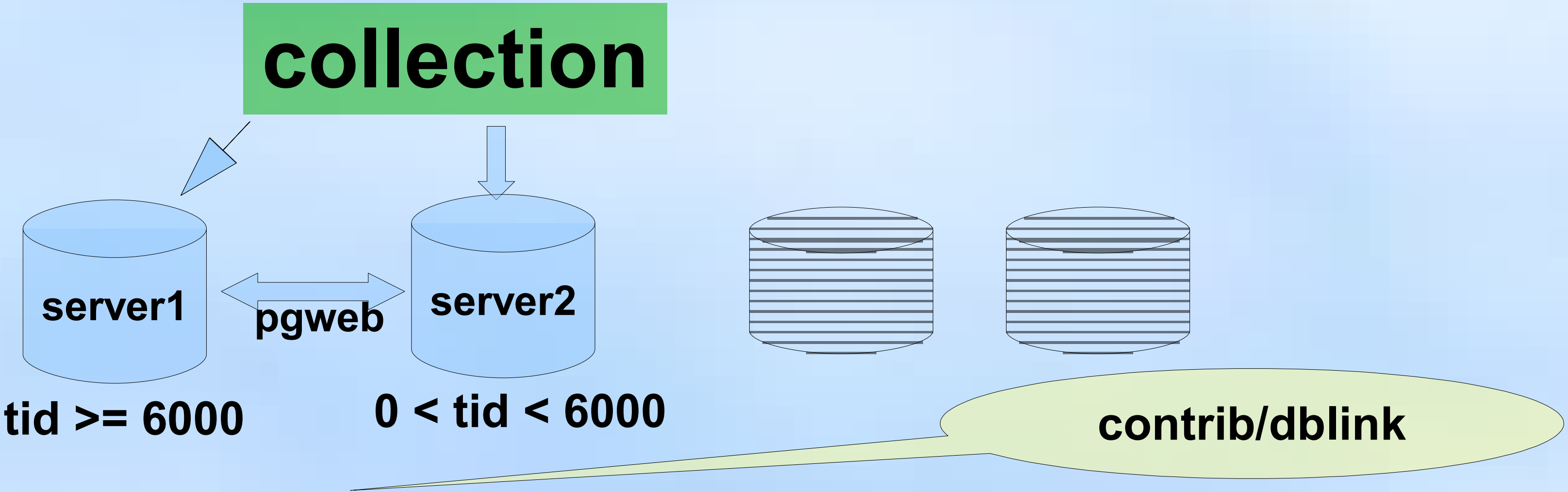
FTS tips – Partition your data

- All queries will not use tables which doesn't match CHECK constraint on `creation_date`

```
arxiv=# explain select title from papers_class where
    fts @@ to_tsquery('stars') and creation_date > '05-01-2007'::date;
                                                 QUERY PLAN
---------------------------------------------------------------------------
 Result  (cost=0.00..63.47 rows=3 width=73)
   ->  Append  (cost=0.00..63.47 rows=3 width=73)
         ->  Seq Scan on papers_class  (cost=0.00..14.95 rows=1 width
               Filter: ((fts @@ '''star'''::tsquery) AND (creation_date
         ->  Bitmap Heap Scan on papers papers_class  (cost=40.53..48.
               Recheck Cond: (creation_date > '2007-05-01'::date)
               Filter: (fts @@ '''star'''::tsquery)
               ->  BitmapAnd  (cost=40.53..40.53 rows=2 width=0)
                     ->  Bitmap Index Scan on gist_idx  (cost=0.00..4.42
                           Index Cond: (fts @@ '''star'''::tsquery)
                     ->  Bitmap Index Scan on creation_date_papers_idx
                           Index Cond: (creation_date > '2007-05-01'::da
```

Big table
paper_archive
was excluded !

FTS tips - Distribute your data



collection

server1

pgweb

server2

tid >= 6000

0 < tid < 6000

contrib/dblink

```
select dblink_connect('pgweb','dbname=pgweb hostaddr='XXX.XXX.XXX.XXX');

select * from dblink('pgweb',
 'select  tid, title, rank_cd(fts_index, q) as rank from pgweb,
    to_tsquery(''table'') q
  where q @@ fts_index and tid >= 6000 order by rank desc limit 10' )
  as t1 (tid integer, title text, rank real)

  union all

  select tid, title,  rank_cd(fts_index, q) as rank from pgweb,
    to_tsquery('table') q
  where q @@ fts_index and tid < 6000 and tid > 0 order by rank desc limit 10

) as foo
order by rank desc limit 10;
```

References

- **Documentation**
  - http://www.sai.msu.su/~megera/postgres/fts/doc - FTSBOOK
  - http://www.sai.msu.su/~megera/wiki/tsearch2 - tsearch2 Wiki
  - http://www.sai.msu.su/~megera/postgres/gist/tsearch/V2 - tsearch2 home page
  - http://www.sai.msu.su/~megera/postgres/talks/ - presentations about PostgreSQL
- **Data**
  - http://www.sai.msu.su/~megera/postgres/fts/apod.dump.gz
- **Acknowledgements**
  - Russian Foundation for Basic Research
  - -hackers, EnterprizeDB PostgreSQL Development Fund, Mannheim University, jfg:networks, Georgia Public Library Service, Rambler Internet Holding

Questions ?

FTS tips

- GIN_FUZZY_SEARCH_LIMIT - maximum number of returned rows
  - GIN_FUZZY_SEARCH_LIMIT=0, disabled on default