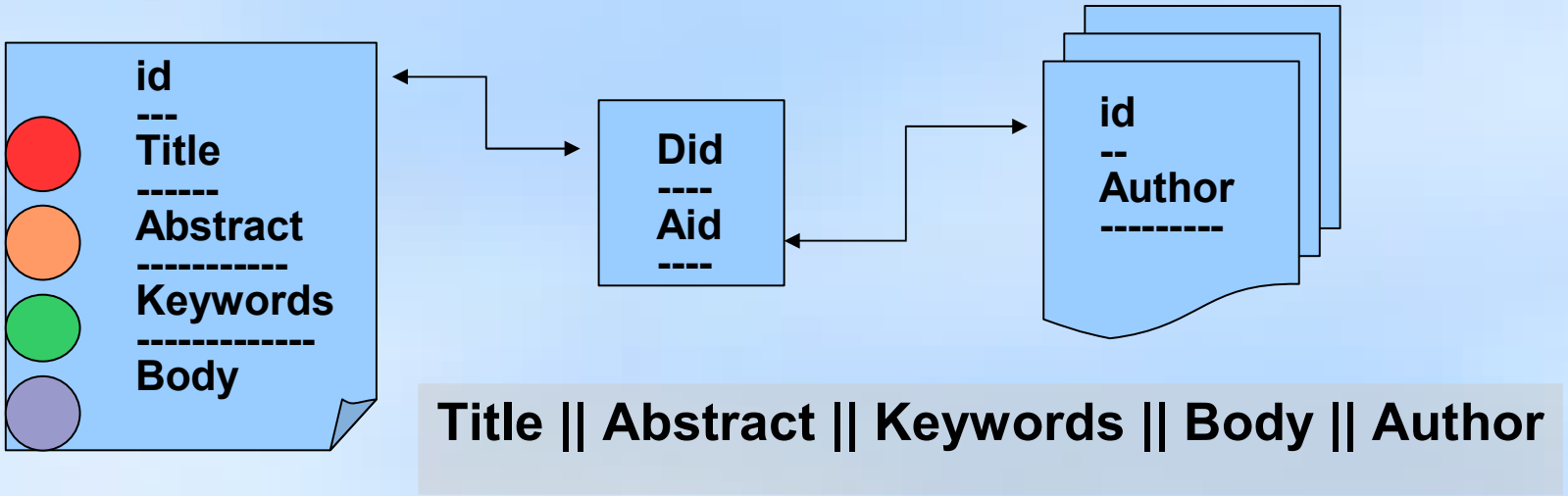

Full-Text Search in PostgreSQL
Oleg Bartunov
Moscow University
PostgreSQL Global Development Group

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



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.

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

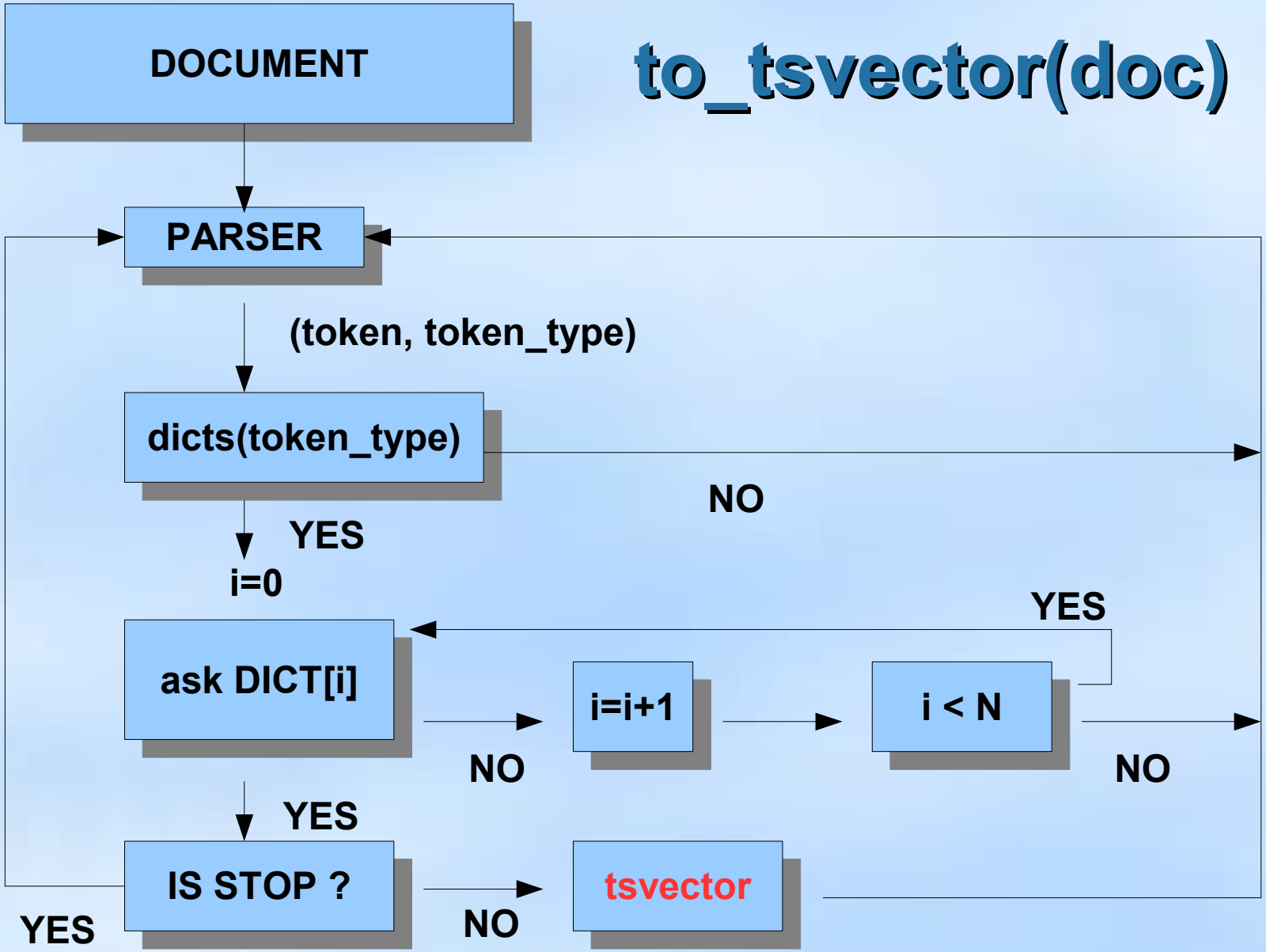
FTS features

- 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

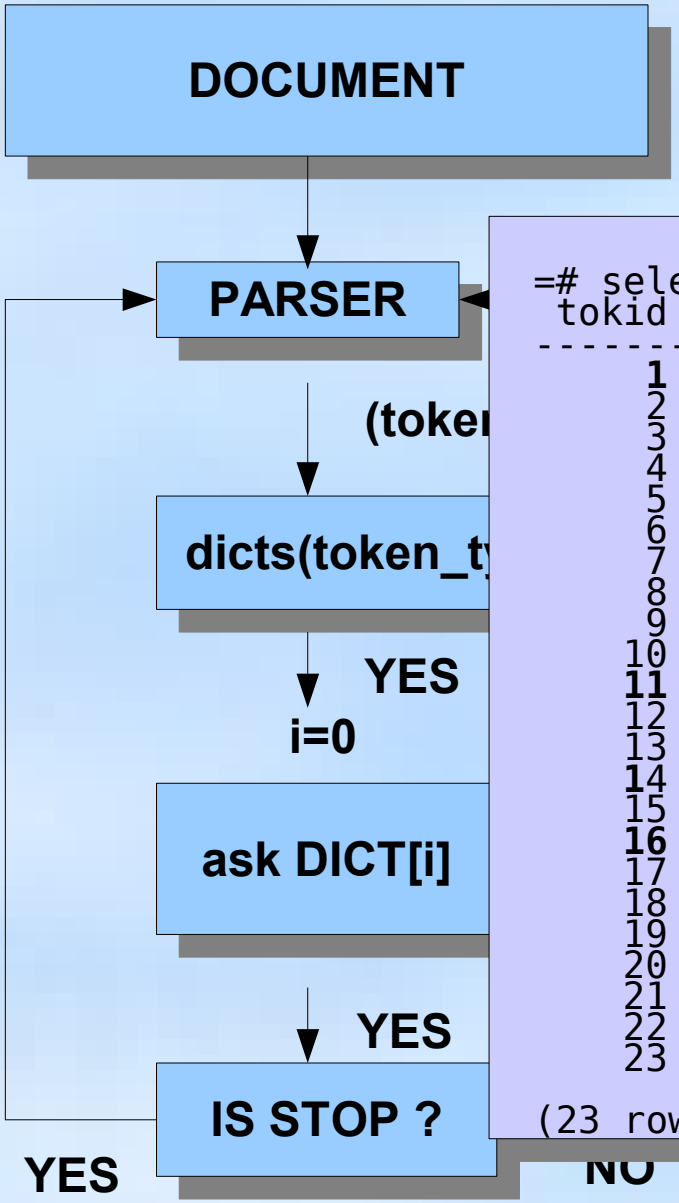
Complete FTS reference

- 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)




```
=# 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)
```

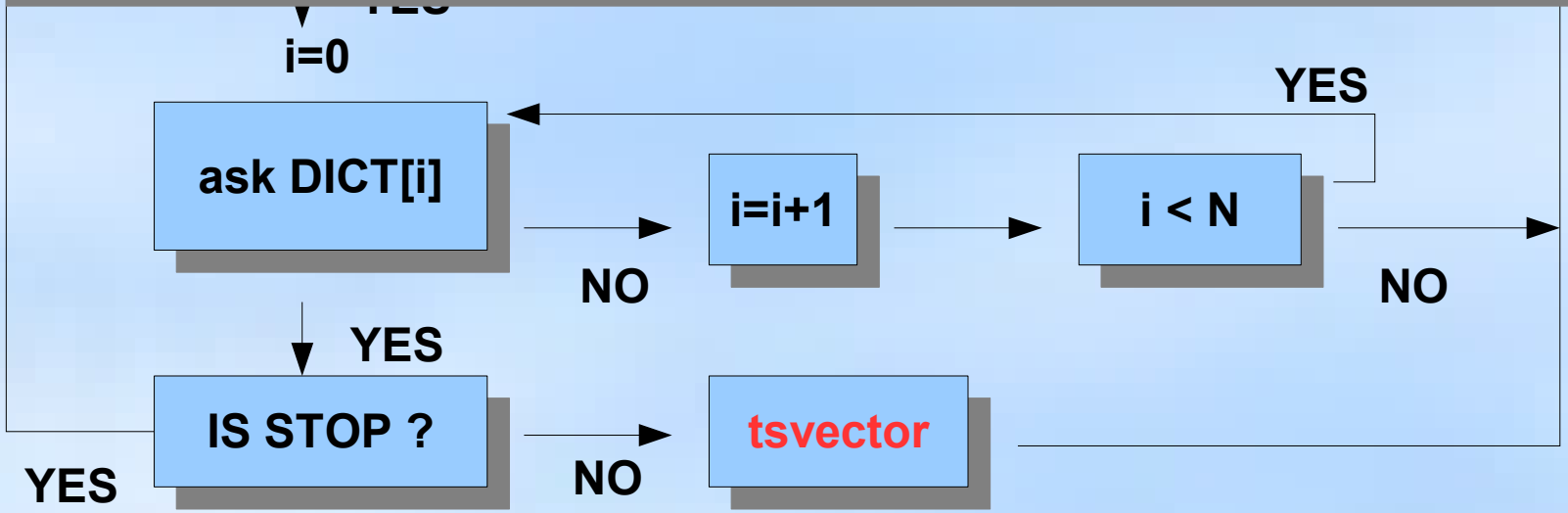
to_tsvector(doc)

DOCUMENT

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

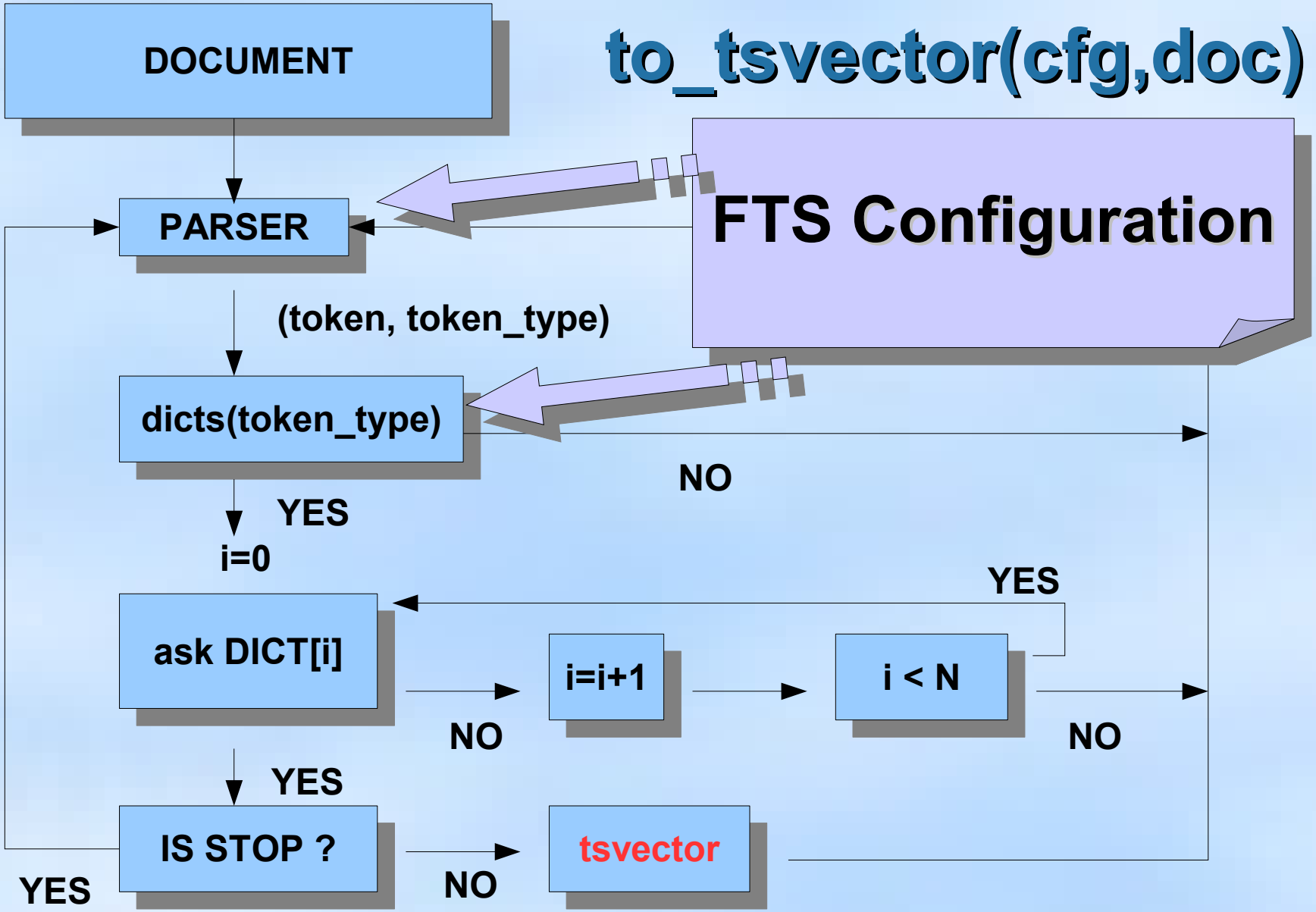

Dictionaries

- Dictionary for roman numerals

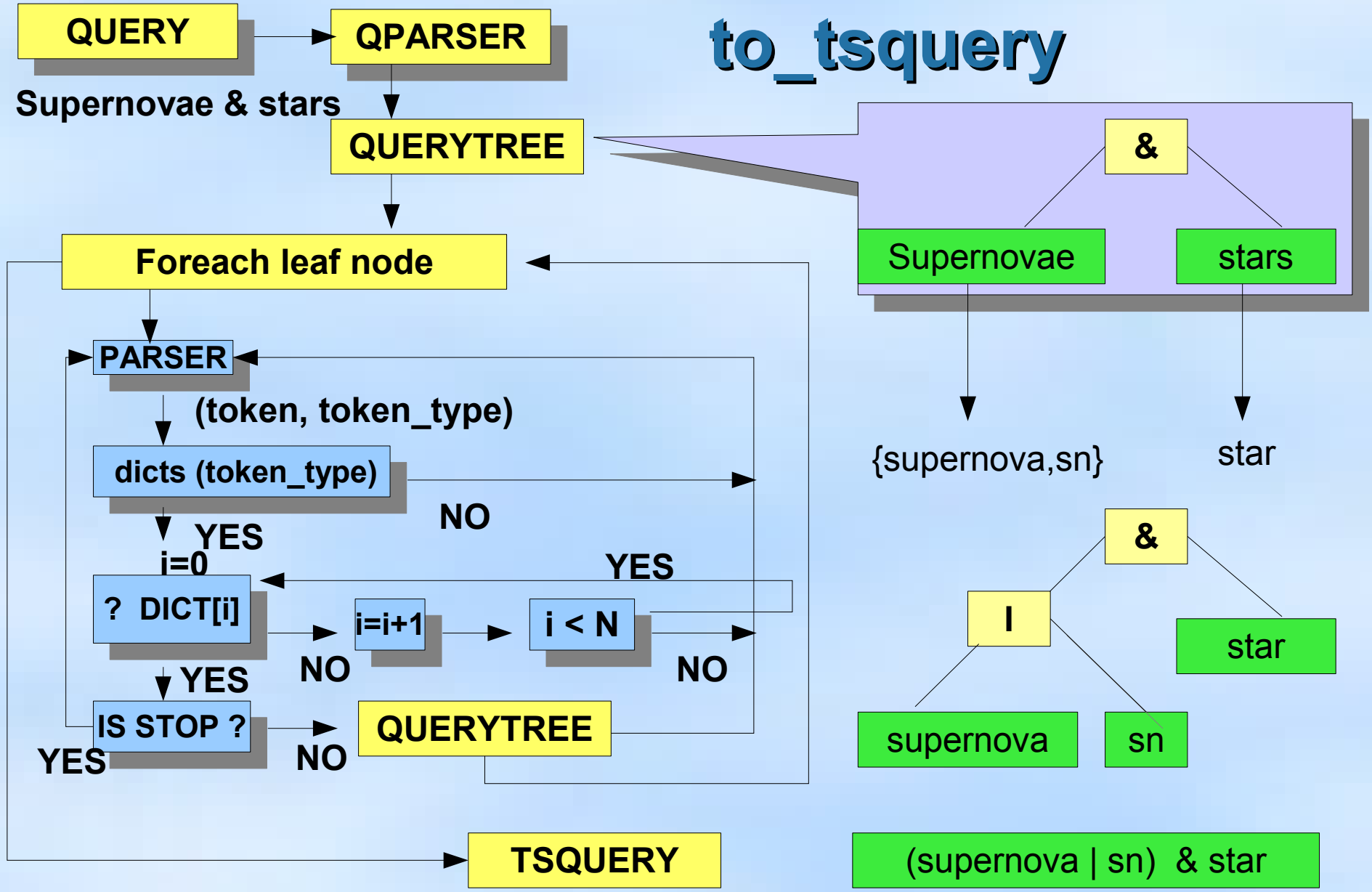
```
=# select lexize('roman', 'XIX');
 lexize
--------
 {19}

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

to_tsvector(cfg,doc)



to_tsquery


QUERY
QPARSER
Supernovae & stars
QUERYTREE
Foreach leaf node
PARSER
(token, token_type)
dicts (token_type)
NO
YES
i=0
YES
? DICT[i]
i=i+1
i < N
NO
NO
YES
IS STOP ?
QUERYTREE
YES
NO
TSQUERY
&
Supernovae
stars
{supernova,sn}
star
&
I
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

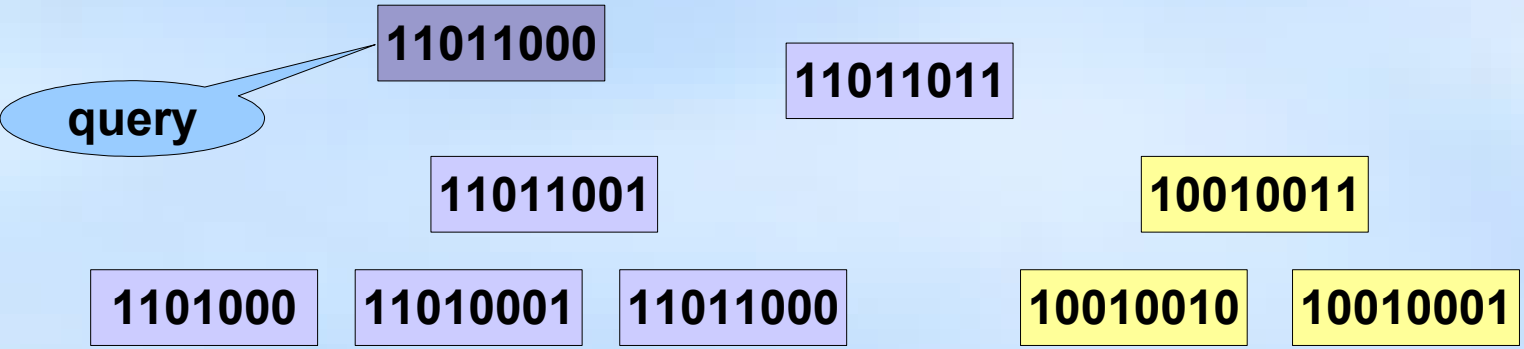
- Query signature – the same way
- Bloom filter

  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 - 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)
```

Leaf node

```
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)
```

Internal node

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

SQL Commands

CREATE FULLTEXT CONFIGURATION -- create full-text configuration

DROP FULLTEXT CONFIGURATION -- remove a full-text configuration

ALTER FULLTEXT CONFIGURATION -- change a full-text configuration

CREATE FULLTEXT DICTIONARY -- create a dictionary for full-text search

DROP FULLTEXT DICTIONARY -- remove a full-text dictionary

ALTER FULLTEXT DICTIONARY -- change a full-text dictionary

CREATE FULLTEXT MAPPING -- binds tokens and dictionaries

ALTER FULLTEXT MAPPING -- change token binding with FTS dictionaries

DROP FULLTEXT MAPPING -- remove mapping

CREATE FULLTEXT PARSER -- create a parser for full-text search

DROP FULLTEXT PARSER -- remove a full-text parser

ALTER FULLTEXT PARSER -- change a full-text parser

ALTER FULLTEXT ... OWNER -- change the owner of a full-text object

COMMENT ON FULLTEXT -- define or change the comment of a full-text object

FTS configuration

```
=# \dF
                                    List of fulltext configurations
   Schema   |         Name          |     Locale      | Default |        Descriptionion
------------+-----------------------+-----------------+---------+------------------------
 pg_catalog | danish_iso_8859_1     | da_DK.ISO8859-1 | Y       |
 pg_catalog | danish_utf_8          | da_DK.UTF-8     | Y       |
 pg_catalog | dutch_iso_8859_1      | nl_NL.ISO8859-1 | Y       |
 pg_catalog | dutch_utf_8           | nl_NL.UTF-8     | Y       |
 pg_catalog | english               | C               | Y       | English
 pg_catalog | finnish_iso_8859_1    | fi_FI.ISO8859-1 | Y       |
 pg_catalog | finnish_utf_8         | fi_FI.UTF-8     | Y       |
 pg_catalog | french_iso_8859_1     | fr_FR.ISO8859-1 | Y       |
 pg_catalog | french_utf_8          | fr_FR.UTF-8     | Y       |
 pg_catalog | german_iso_8859_1     | de_DE.ISO8859-1 | Y       |
 pg_catalog | german_utf_8          | de_DE.UTF-8     | Y       |
 pg_catalog | hungarian_iso_8859_1  | hu_HU.ISO8859-1 | Y       |
 pg_catalog | hungarian_utf_8       | hu_HU.UTF-8     | Y       |
 pg_catalog | italian_iso_8859_1    | it_IT.ISO8859-1 | Y       |
 pg_catalog | italian_utf_8         | it_IT.UTF-8     | Y       |
 pg_catalog | norwegian_iso_8859_1  | no_NO.ISO8859-1 | Y       |
 pg_catalog | norwegian_utf_8       | no_NO.UTF-8     | Y       |
 pg_catalog | portuguese_iso_8859_1 | pt_PT.ISO8859-1 | Y       |
 pg_catalog | portuguese_utf_8      | pt_PT.UTF-8     | Y       |
 pg_catalog | russian_koi8          | ru_RU.KOI8-R    | Y       | Russian/KOI8-R
 pg_catalog | russian_utf8          | ru_RU.UTF-8     | Y       | Russian/UTF-8
 pg_catalog | russian_win1251       | ru_RU.CP1251    | Y       | Russian/WIN-1251
 pg_catalog | simple                |                 |         | Simple configuration
 pg_catalog | spanish_iso_8859_1    | es_ES.ISO8859-1 | Y       |
 pg_catalog | spanish_utf_8         | es_ES.UTF-8     | Y       |
 pg_catalog | swedish_iso_8859_1    | sv_SE.ISO8859-1 | Y       |
 pg_catalog | swedish_utf_8         | sv_SE.UTF-8     | Y       |
(27 rows)
```

27 configurations for 10 languages armed with snowball stemmers

FTS configuration

- **Many FTS configurations, but only one default in schema**
- **Servers locale** defines default FTS configuration
  - show lc_ctype;
  - show lc_collate;
- GUC variable **tsearch_conf_name** contains name of active FTS configuration;
  - define in postgresql.conf
  - set **tsearch_conf_name**=simple;
  - alter user httpd set **tsearch_conf_name=simple;**
- **search_path** defines an order of schema to search FTS configuration
  - **pg_catalog implicitly placed first in search_path**

    pg_catalog,"$user",public
  - **set search_path=public,pg_catalog;**
- Use schema.qualified name of FTS configuration if unsure

SQL Commands: FTS configuration

SQL:

```
CREATE FULLTEXT CONFIGURATION cfgname
    PARSER  prsname  [ LOCALE localename]
[AS DEFAULT];

CREATE FULLTEXT CONFIGURATION cfgname
  [{  PARSER  prsname | LOCALE  localename  }  [ ...]]
LIKE template_cfg [WITH MAP]
[AS DEFAULT];
```

FUNCTIONAL:

```
SELECT fts_cfg_create( cfgname name, template_cfg name ,prsname
name, localename text,  with_map bool, default bool);
```

OLD:

```
INSERT INTO pg_ts_cfg VALUES(cfgname,prsname,localename);

INSERT INTO pg_ts_cfgmap  SELECT  cfgname, tok_alias, dict_name
FROM pg_ts_cfgmap WHERE  ts_name=template_cfg;
```

SQL Commands: FTS dictionary

```
CREATE FULLTEXT DICTIONARY dictname
    LEXIZE  lexize_function
    [INIT  init_function ]
    [OPTION  opt_text ]
;
CREATE FULLTEXT DICTIONARY dictname
[    {   INIT  init_function
       | LEXIZE  lexize_function
       | OPTION opt_text }
[ ... ]] LIKE template_dictname;


=# CREATE FULLTEXT DICTIONARY public.my_simple OPTION
   'english.stop' LIKE pg_catalog.simple;
```


SQL Commands: FTS mapping

CREATE FULLTEXT MAPPING ON cfgname

FOR tokentypename[, ...] WITH dictname1[, ...];

```
=# CREATE FULLTEXT MAPPING ON testcfg FOR
   lword,lhword,lpart_hword  WITH simple,en_stem;
=# \dF+ testcfg
Configuration 'testcfg'
Parser name: 'default'
Locale: 'testlocale'
     Token    |  Dictionaries
--------------+----------------
 lhword       | simple,en_stem
 lpart_hword  | simple,en_stem
 lword        | simple,en_stem
```

Pgweb example

$PGROOT/share

transaction !

```
BEGIN;

CREATE FULLTEXT CONFIGURATION public.pg  LOCALE  'ru_RU.UTF-8' LIKE english WITH MAP
                                                                    AS DEFAULT;
CREATE FULLTEXT DICTIONARY pg_dict OPTION 'pg_dict.txt'  LIKE synonym;
CREATE FULLTEXT DICTIONARY en_ispell
    OPTION 'DictFile="/usr/local/share/dicts/ispell/english-utf8.dict",
            AffFile="/usr/local/share/dicts/ispell/english-utf8.aff",
            StopFile="/usr/local/share/dicts/ispell/english-utf8.stop"'
    LIKE ispell_template;
ALTER FULLTEXT DICTIONARY en_stem SET OPTION '/usr/local/share/dicts/ispell/english-utf8.stop';
ALTER FULLTEXT MAPPING ON pg FOR lword,lhword,lpart_hword
                        WITH pg_dict,en_ispell,en_stem;
DROP FULLTEXT MAPPING ON pg FOR email, url, sfloat, uri, float;
END;
```

postgres postgresql
pgsql postgresql
postgre postgresql

don't index email, url,sfloat,uri,float

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
```

Default configuration for **ru_RU.UTF-8** locale

APOD example: FTS configuration

```
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: obtaining FTS index

```
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 ''

A > B > D

```
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 |
```

collect statistics

```
=# 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
```

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 fulltext dictionary **en_stem_nostop** OPTION NULL like **en_stem**;
create fulltext configuration **hamlet** LIKE english WITH MAP;
alter fulltext mapping on hamlet for 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:
(1 row)
```

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**

Low limit

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;
```

new document

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;
```

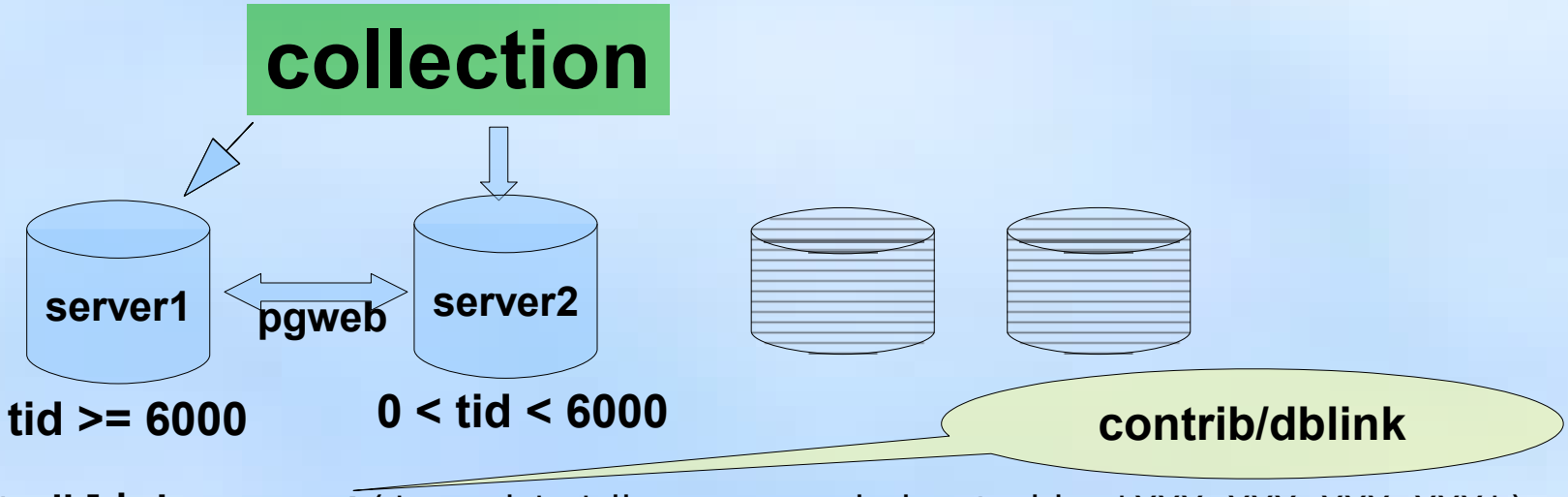
title	rank
New Shocks For Supernova 1987A	**0.533312**

Old rank

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 - Distribute your data




```
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;
```

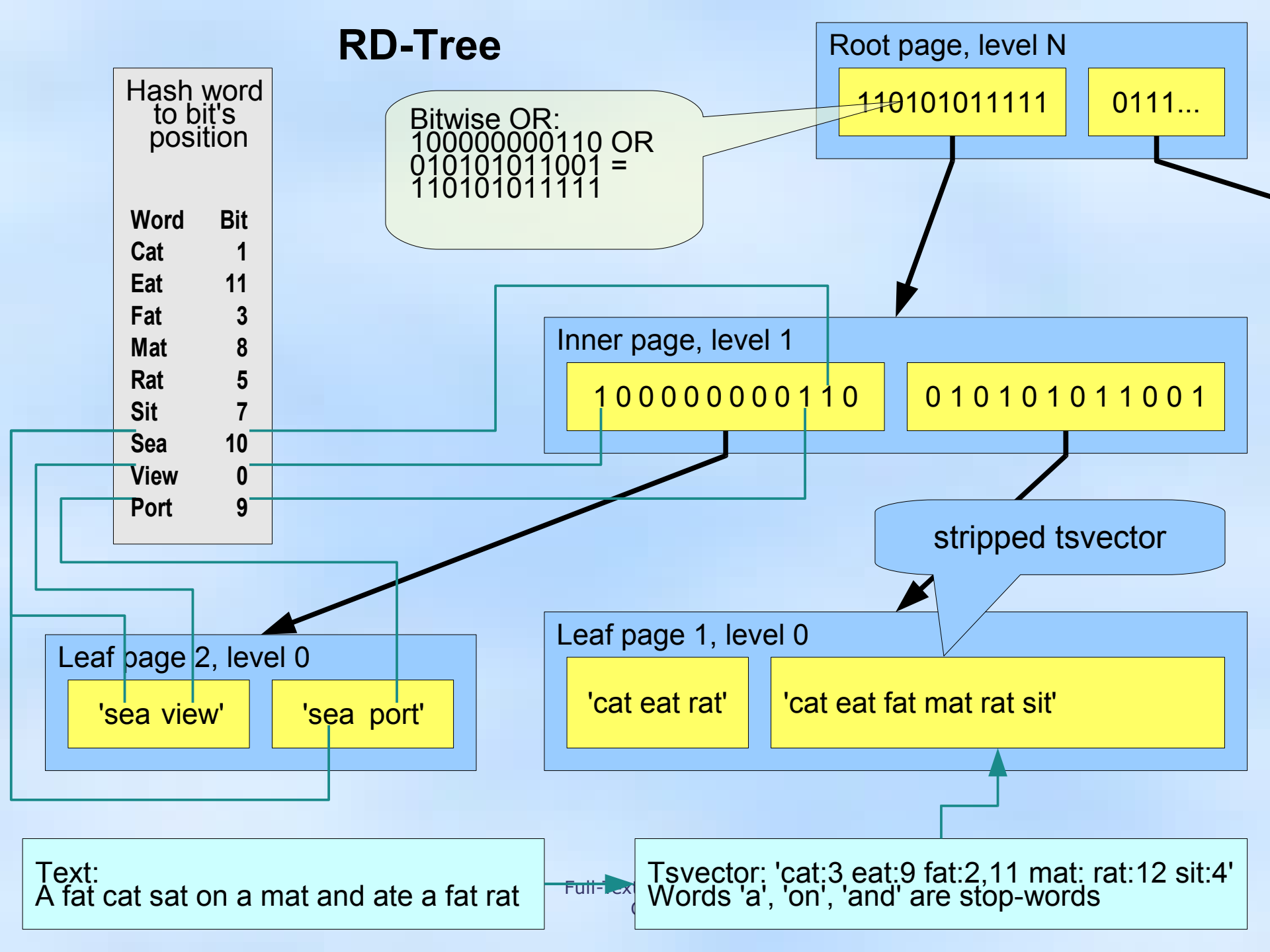
References

- **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

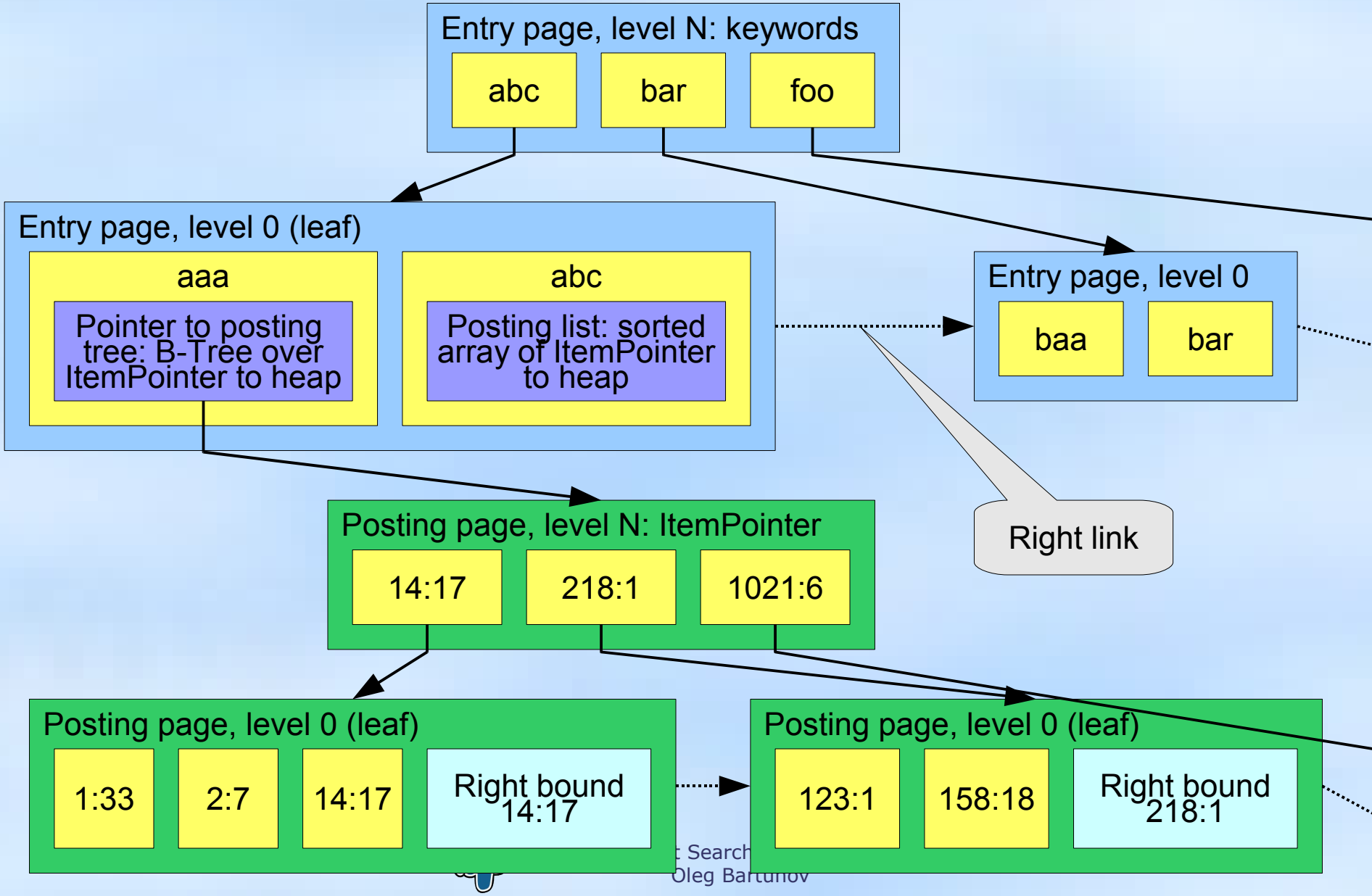
Questions ?

FTS tips

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


RD-Tree
Hash word to bit's position
Word Bit
Cat 1
Eat 11
Fat 3
Mat 8
Rat 5
Sit 7
Sea 10
View 0
Port 9
Bitwise OR:
100000000110 OR
010101011001 =
110101011111
Root page, level N
110101011111
0111...
Inner page, level 1
1 0 0 0 0 0 0 0 0 1 1 0
0 1 0 1 0 1 0 1 1 0 0 1
stripped tsvector
Leaf page 2, level 0
'sea view'
'sea port'
Leaf page 1, level 0
'cat eat rat'
'cat eat fat mat rat sit'
Text:
A fat cat sat on a mat and ate a fat rat
Full-Tex
Tsvector: 'cat:3 eat:9 fat:2,11 mat: rat:12 sit:4'
Words 'a', 'on', 'and' are stop-words

GIN


Entry page, level N: keywords
abc
bar
foo
Entry page, level 0 (leaf)
aaa
Pointer to posting tree: B-Tree over ItemPointer to heap
abc
Posting list: sorted array of ItemPointer to heap
Entry page, level 0
baa
bar
Right link
Posting page, level N: ItemPointer
14:17
218:1
1021:6
Posting page, level 0 (leaf)
1:33
2:7
14:17
Right bound 14:17
Posting page, level 0 (leaf)
123:1
158:18
Right bound 218:1