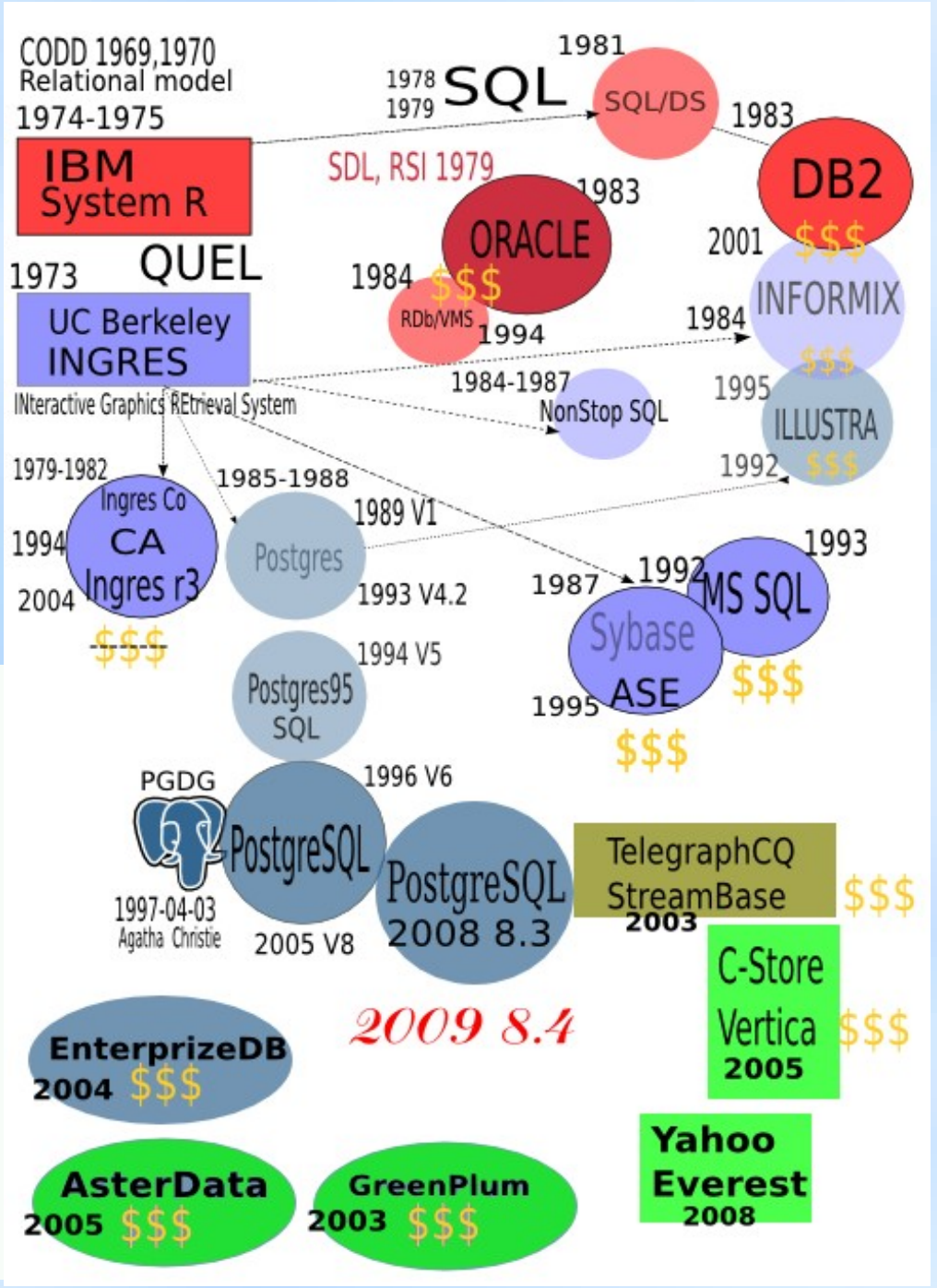
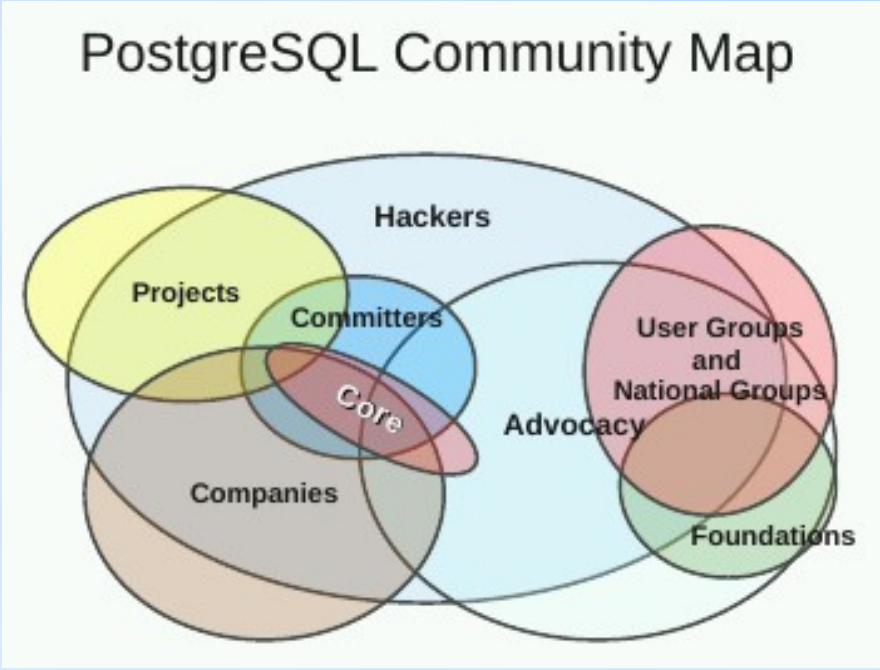

Full-text search in PostgreSQL
Oleg Bartunov
Moscow University, SAI
PostgreSQL Global Development Group


I. What is PostgreSQL
II. Full-text search in PostgreSQL

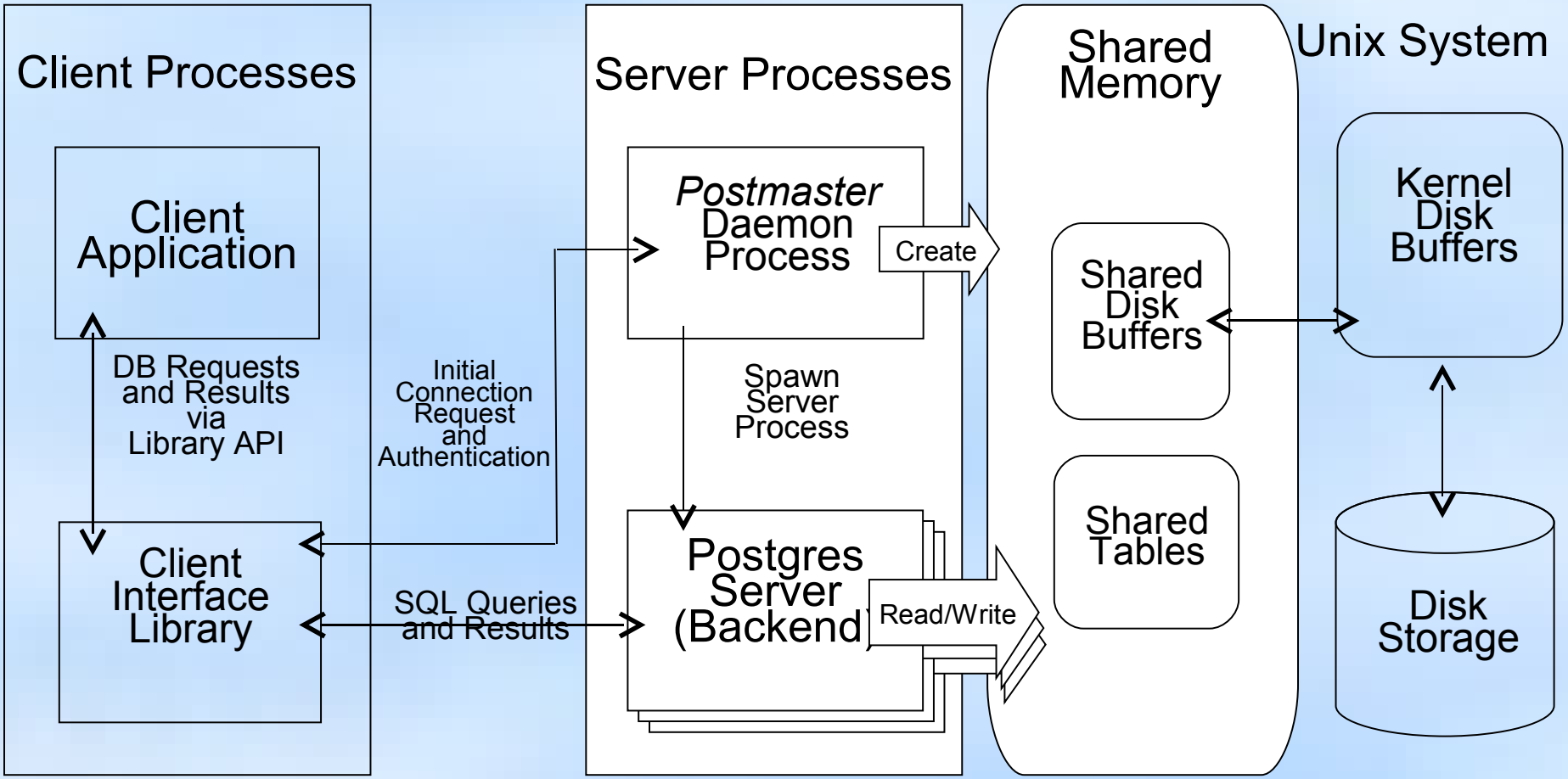
Pedigree of PostgreSQL

- Large distributed user and developer community
- Not owned by any one company
- «Community owned»
- About 200 developers in 14 time zones



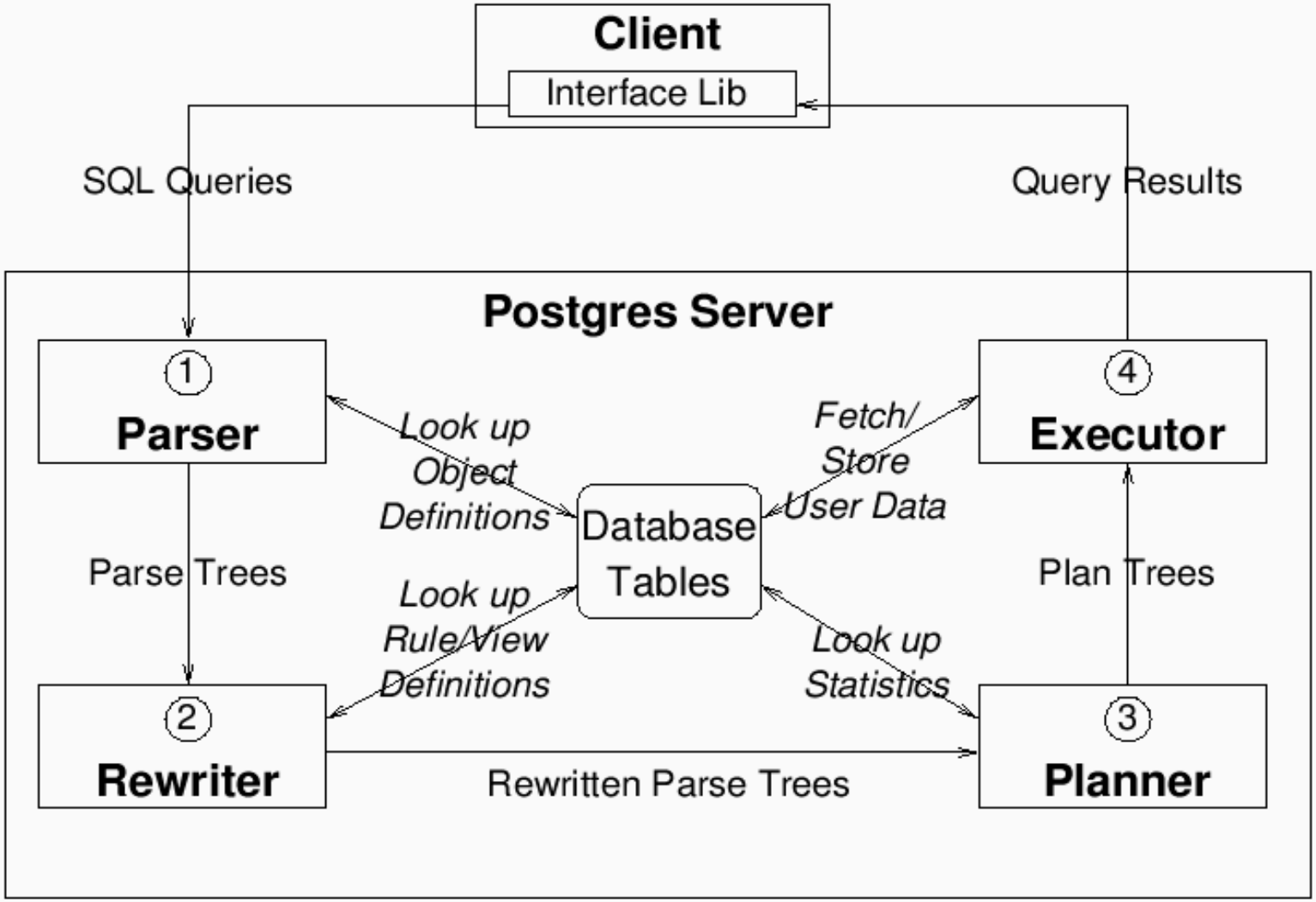


Architecture



Server processes: **bgwriter** (dirty shared buffers ->disk), **statistic collector**, **autovacuum launcher**

Query Processing



Key data structures: parse tree, plan tree

Babel of PLs

- SQL,C,C++
- PL/pgSQL
- PL/Perl
- PL/Python
- PL/Tcl
- PL/Java, PL/Py, PL/PHP, PL/R, PL/Ruby, PL/Scheme, PL/sh, PL/Lua, PL/Parrot
- **Trusted** — superuser, open pipes, filehandles, **Untrusted** — regular user

Migration

- Closest to proprietary enterprise Dbs
- Automatic migration from Informix
  - Informix is ~ 50% PostgreSQL
- Relatively easy migration from Oracle
  - EnterpriseDB provides Pl/Psql -> PL/PgSQL
- SQL Server, DB2 harder
  - But easier than MySQL

Feature	ASE	DB2	FireBird	InterBase	MS SQL	MySQL	Oracle	PostgreSQL
License	$$$	$$$	IPL[2]	$$$	$$$	GPL/$$$	$$$	BSD
ACID	Yes	Yes	Yes	Yes	Yes	Depends[1]	Yes	Yes
Referential integrity	Yes	Yes	Yes	Yes	Yes	Depends[1]	Yes	Yes
Transaction	Yes	Yes	Yes	Yes	Yes	Depends[1]	Yes	Yes
Unicode	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Schema	Yes	Yes	Yes	Yes	No[5]	No	Yes	Yes
Temporary table	No	Yes	No	Yes	Yes	Yes	Yes	Yes
View	Yes	Yes	Yes	Yes	Yes	No	Yes	Yes
Materialized view	No	Yes	No	No	No	No	Yes	No[3]
Expression index	No	No	No	No	No	No	Yes	Yes
Partial index	No	No	No	No	No	No	Yes	Yes
Inverted index	No	No	No	No	No	Yes	Yes	Yes
Bitmap index	No	Yes	No	No	No	No	Yes	No
Domain	No	No	Yes	Yes	No	No	Yes	Yes
Cursor	Yes	Yes	Yes	Yes	Yes	No	Yes	Yes
User Defined Functions	Yes	Yes	Yes	Yes	Yes	No[4]	Yes	Yes
Trigger	Yes	Yes	Yes	Yes	Yes	No[4]	Yes	Yes
Stored procedure	Yes	Yes	Yes	Yes	Yes	No[4]	Yes	Yes
Tablespace	Yes	Yes	No	?	No[5]	No[1]	Yes	Yes
Название	**ASE**	**DB2**	**FireBird**	**InterBase**	**MS SQL**	**MySQL**	**Oracle**	**PostgreSQL**

Remarks:

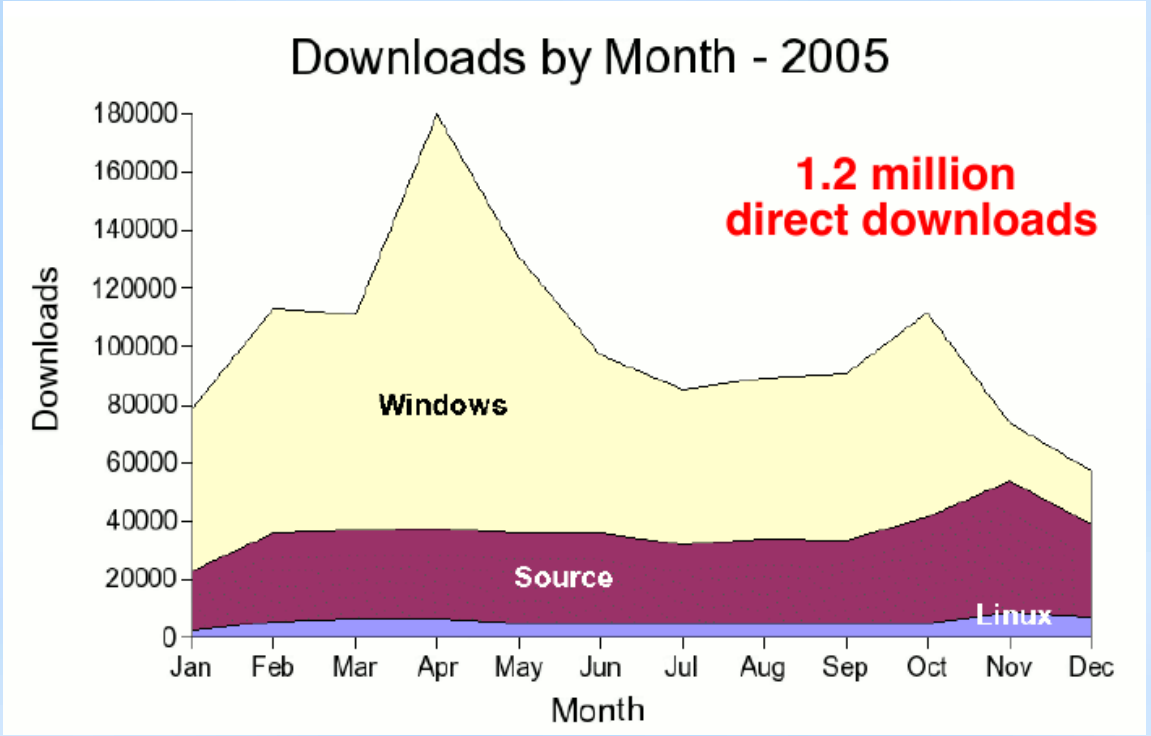
- **1** - only InnoDB (not default)
- **2** - Interbase Public License
- **3** - Materialized view emulated using PL/pgSQL
- **4** - only MySQL 5.0
- **5** - only MS SQL Server 2005 (Yukon)

Limitations

- Maximal DB size – unlimited
- Maximal Table size – 32Тб
- Maximal tuple length – 400Gb
- Maximal attribute length – 1 Gb
- Maximal number of rows – unlimited
- Maximal number of columns – 250-1600
- Maximal number of indexes - unlimited

PostgreSQL Release Policy

- Time-based releases
  - No new features in minor releases
  - Every year
    - 7.4: 2003
    - 8.0: 2004
    - 8.1: 2005
    - 8.2: 2006
    - 8.3: 2008
    - 8.4: 2009



PostgreSQL BuildFarm Status

Shown here is the latest status of each farm member for each branch it has reported on in the last 30 days.

Use the farm member link for history of that member on the relevant branch.

Legend	= cassert	= debug	= integer-datetimes	= krb5	= nls	= openssl
	= pam	= perl	= python	[tcl] = tcl	= thread-safety	= xml

Branch: HEAD			
Alias	**System**	**Status**	**Flags**
koi	unixware 7.1.4 sco cc 4.2 bi-xeon HT	00:24:14 ago OK Config	[tcl]
tapir	FreeBSD 6.0 gcc 3.4.4 amd64	00:25:13 ago OK Config	
marten	CentOS 5 GCC 4.1.2 x86	00:25:20 ago OK Config	[tcl]
narwhal	Windows Server 2003 R2 5.2.3790 GCC 3.4.2 (mingw-special) i686	00:30:13 ago OK Config	[tcl]
mastodon	Windows Server 2003 R2 5.2.3790 MSVC 2005 Express 8.0.50727.42 x86	01:30:12 ago OK Config	[tcl]
dungbeetle	Fedora Core 6 gcc 4.1.1 x86_64	01:46:12 ago OK Config	[tcl]
bear	Tru64 5.0 cc V6.1-011 alpha	02:24:12 ago OK Config	
centaur	CentOS CentOS 5.2 gcc gcc 4.1.2 Athlon MP	02:24:13 ago OK Config	[tcl]
dawn_bat	Windows XP-Pro SP2 gcc 3.4.2 i386	02:30:11 ago OK Config	[tcl]
guanaco	CentOS 4 GCC 3.4.6 x86	02:30:12 ago OK Config	[tcl]
red_bat	Windows XP-PRO SP2 MSVC++ 2005 Express i686	04:00:12 ago OK Config	[tcl]
wombat	Gentoo 1.12.9 gcc 4.1.1 ppc64 PPC970MP	04:00:13 ago OK Config	[tcl]

Top10 reasons to use PostgreSQL

- **Reliability**: 15 year of development, smart and active community
- **Security**: Most secure major SQL RDBMS
- **Advanced SQL**: Supports complex queries and ANSI SQL 2003
- **Data Integrity**: Never loose a transaction or violate a key, ever (full ACID)
- **Extensibility**: GIS, genomics, XML, cryptography, full-text search and more

Top10 reasons to use PostgreSQL

- **Languages**: Interface with or write stored procedures in any languages
- **Hackability**: Clear, readable, modifiable source code
- **Freedom**: BSD-licensed to use for any purpose, include commercia
- **Performance**: Equivalent to top proprietary RDBMS
- **Size**: Easily supports multi-terabyte db

PostgreSQL conferences

- PGCon Canada — international conference
- PG Conferences U.S. (East,West) - US conferences
- PgDay.FR — a French conference
- PgDay.IT — European conference
- PgCon Brazil — The Brazilian PostgreSQL Conferences
- Also, Pg sections in all major worldwide OSS conferences !

Development Priorities (historical)

PostgreSQL

1. Data Integrity
2. Security
3. Reliability
4. Standards
5. DB Features
6. Performance
7. Easy-of-use
8. Programmer Features

MySQL

1. Easy-of-use
2. Performance
3. Programmer Features
4. Reliability
5. DB Features
6. Data Integrity
7. Security
8. Standards

Summer 2007: The 1st PostgreSQL Enterprise-level Benchmark (SPEC*)

Josh Berkus: «...a good day for Open Source»

PostgreSQL 8.2 – 813.73 JOPS

- SPECjAppServer2004 2x Sun Fire X4200 appservers (8 cores, 4 chips) and 1 Sun Fire T2000 DB server (8 cores, 1 chips) with PostgreSQL 8.2.4
- HW: ~**$65,000**; SW: **$0**

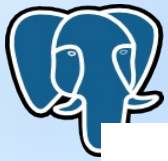
Oracle 10g – 874.17 JOPS

- SPECjAppServer2004 1 HP rx2660 appserver (4 cores, 2 chips) and 1 rx2660 DB server (4 cores, 2 chips) with Oracle Database 10g Enterprise Edition Release 10.2.0.2
- HW: **~$74,000**; SW: **~$110,000**

$118500 economy/per server without loss of performance !

---

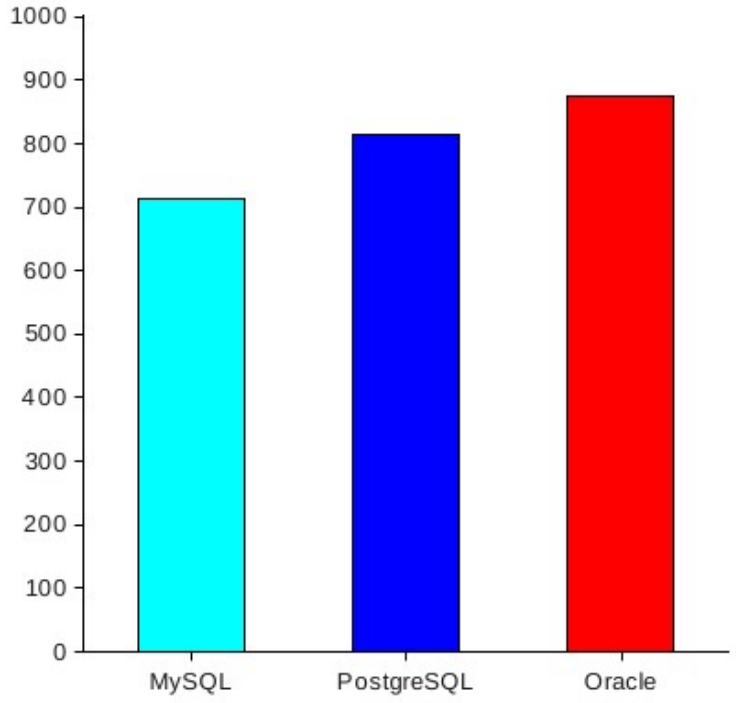
*) SPEC – Standard Performance Evaluation Corporation, http://spec.org



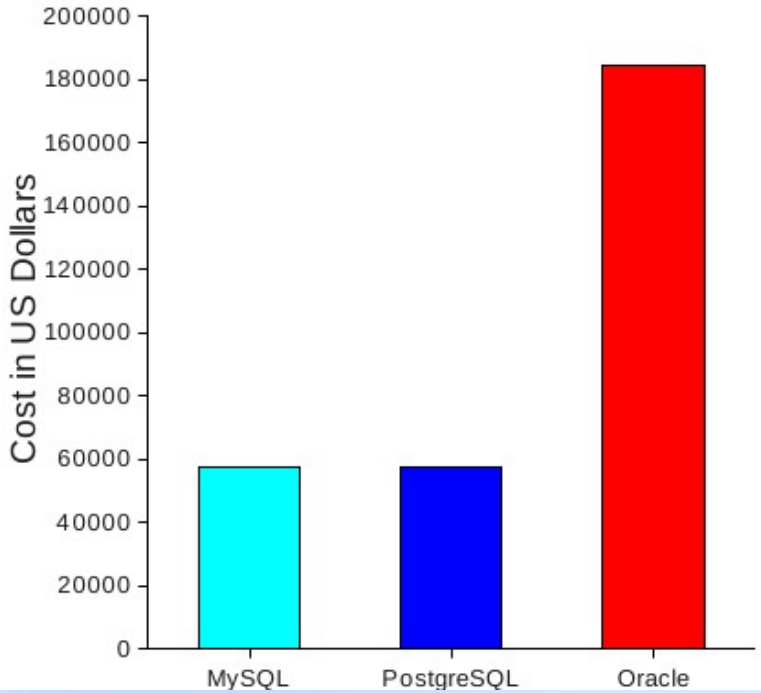
Benchmarks

- SpecJAppserver 2004, as of July 2007

J2EE Througput



Acquisition Cost Comparison



PostgreSQL news

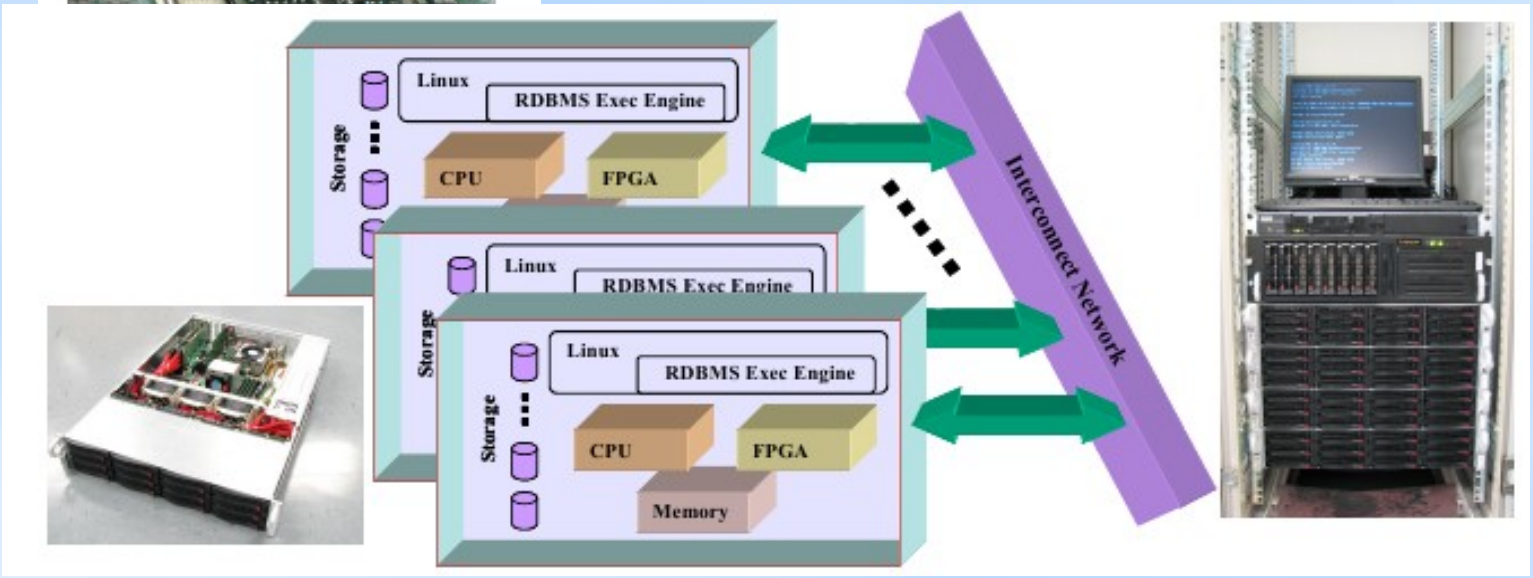
- EnterpriseDB 2008
  - 10 mln. $$$ investments from IBM
  - No longer slogan «Oracle compatibility»
  - Rebranding EnterpriseDB
    - Postgres Plus — A true Enterprise Open Source DB
    - GridSQL — execution in Grid environment
    - Postgres Plus Advanced Server — commercial
      - Cloud Edition (Elastra)
      - Oracle compatibility, migration
      - DynaTune — autotuning

PostgreSQL news

- XDI Applience — postgreSQL with support of shared-nothing parallel cluster envrionment with FPGA acceleration
  - 1GB/s SQL query per Node (TPC-H)
  - 1TB/min sustained SQL query 19 inch server rack
  - Decision Support Systems, Full table scan, Group By, Order By, Aggregation, multitable joins

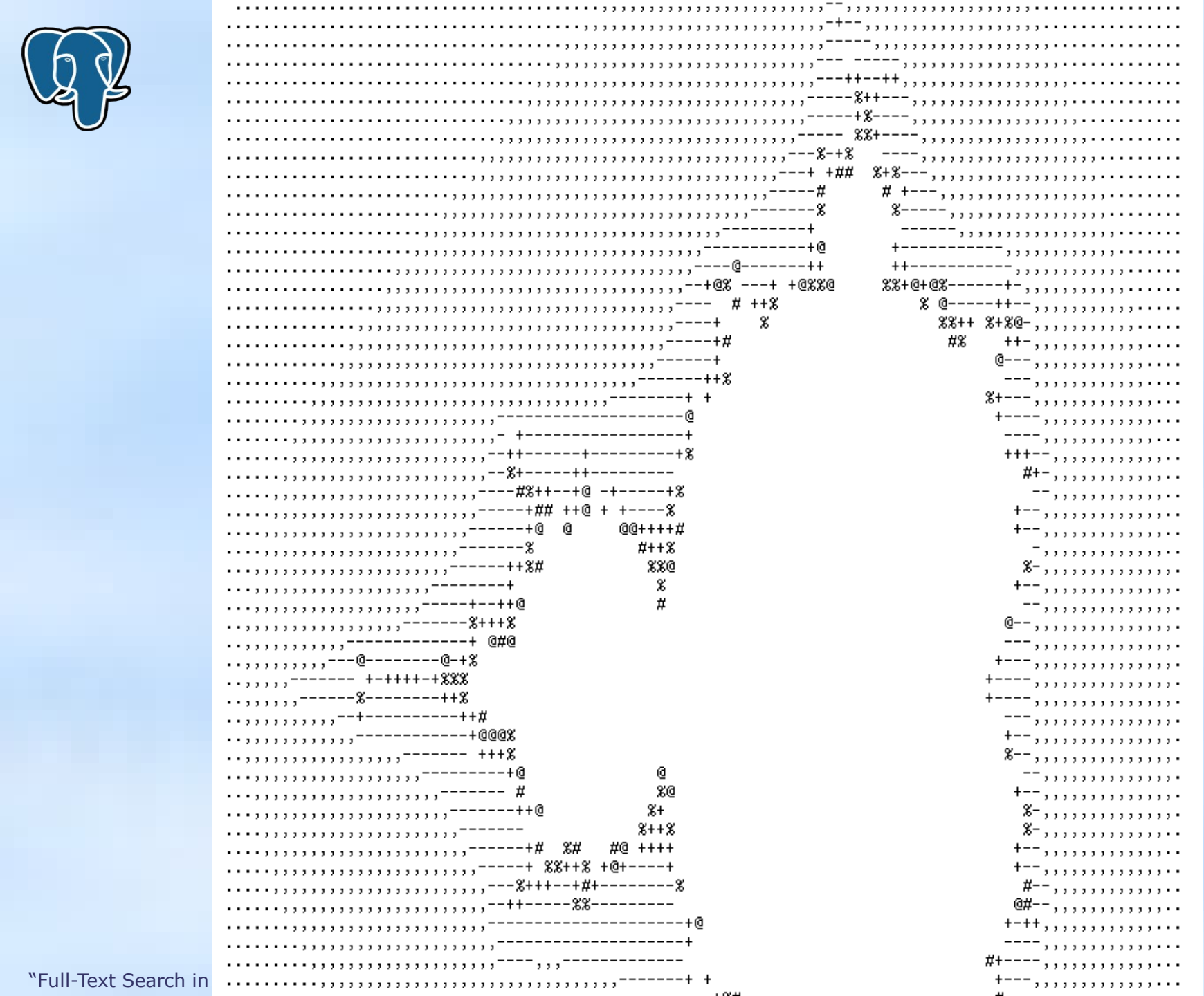
PostgreSQL news

- XDI Appliance — 750,000 USD



Mandelbrot with SQL

```
Time: 369.131 ms
postgres=# WITH RECURSIVE
x(i)
AS (
    VALUES(0)
UNION ALL
    SELECT i + 1 FROM x WHERE i < 101
),
Z(Ix, Iy, Cx, Cy, X, Y, I)
AS (
    SELECT Ix, Iy, X::float, Y::float, X::float, Y::float, 0
    FROM
        (SELECT -2.2 + 0.031 * i, i FROM x) AS xgen(x,ix)
    CROSS JOIN
        (SELECT -1.5 + 0.031 * i, i FROM x) AS ygen(y,iy)
    UNION ALL
    SELECT Ix, Iy, Cx, Cy, X * X - Y * Y + Cx AS X, Y * X * 2 + Cy, I + 1
    FROM Z
    WHERE X * X + Y * Y < 16.0
    AND I < 27
),
Zt (Ix, Iy, I) AS (
    SELECT Ix, Iy, MAX(I) AS I
    FROM Z
    GROUP BY Iy, Ix
    ORDER BY Iy, Ix
)
SELECT array_to_string(
    array_agg(
        SUBSTRING(
            ' .,,,-----++++%%%%@@@@#### ',
            GREATEST(I,1),
            1
        )
    ),''
)
FROM Zt
GROUP BY Iy
ORDER BY Iy;
```



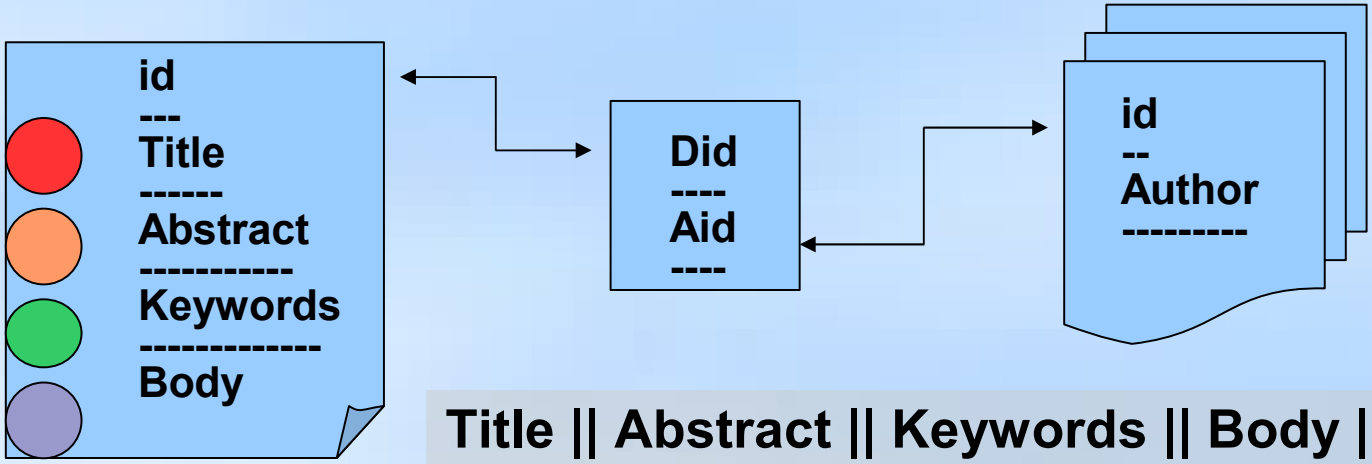

Full-text search in
PostgreSQL

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



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**

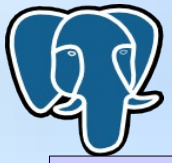
  tsvector @@ tsquery

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

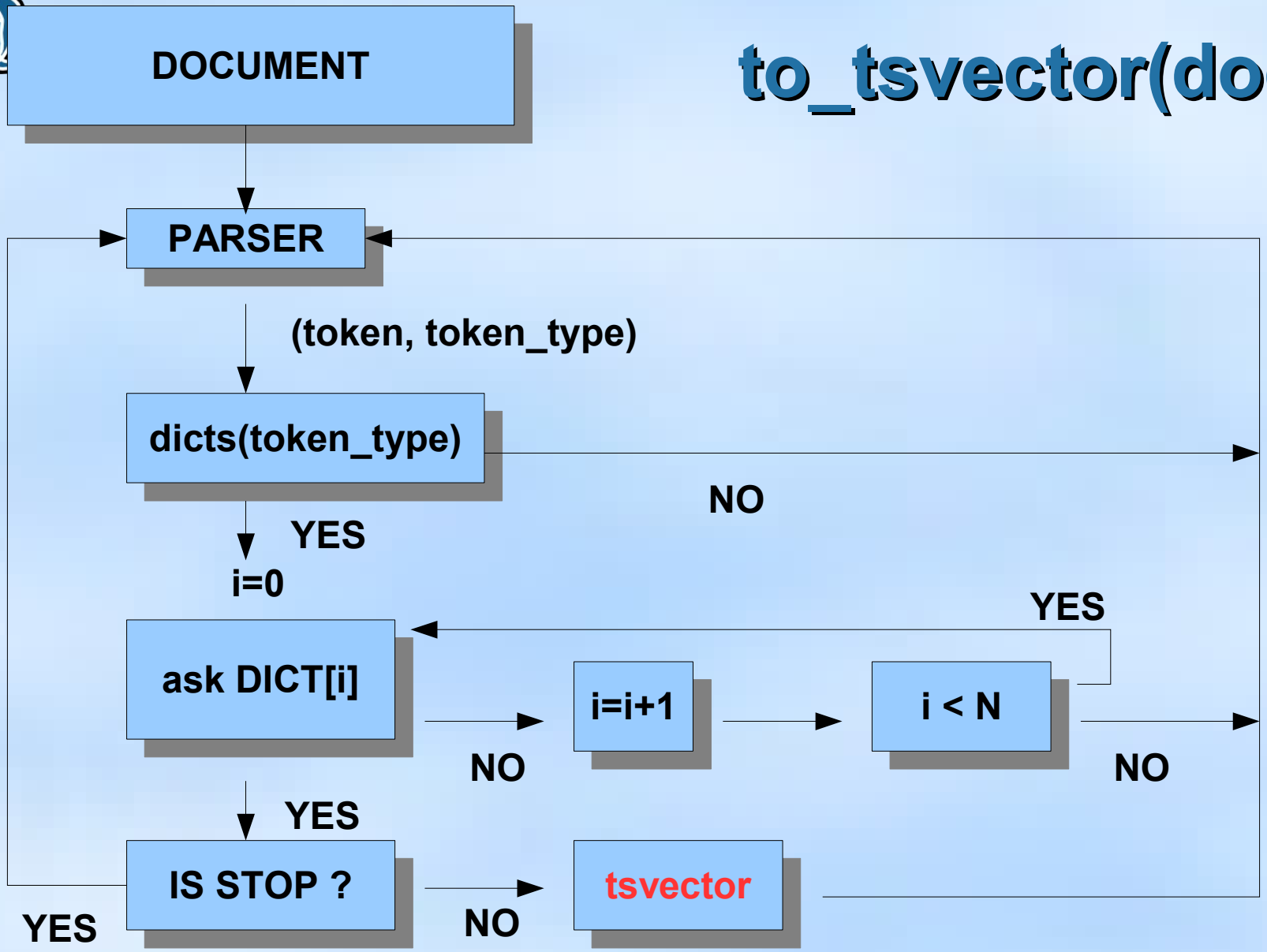
- Full integration with PostgreSQL
- 27 built-in configurations for 10 languages
- Support of user-defined FTS configurations
- Pluggable dictionaries (ispell, snowball, thesaurus), parsers
- Multibyte support (UTF-8)
- Relevance ranking
- Two types of indexes – GiST and GiN with concurrency and recovery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



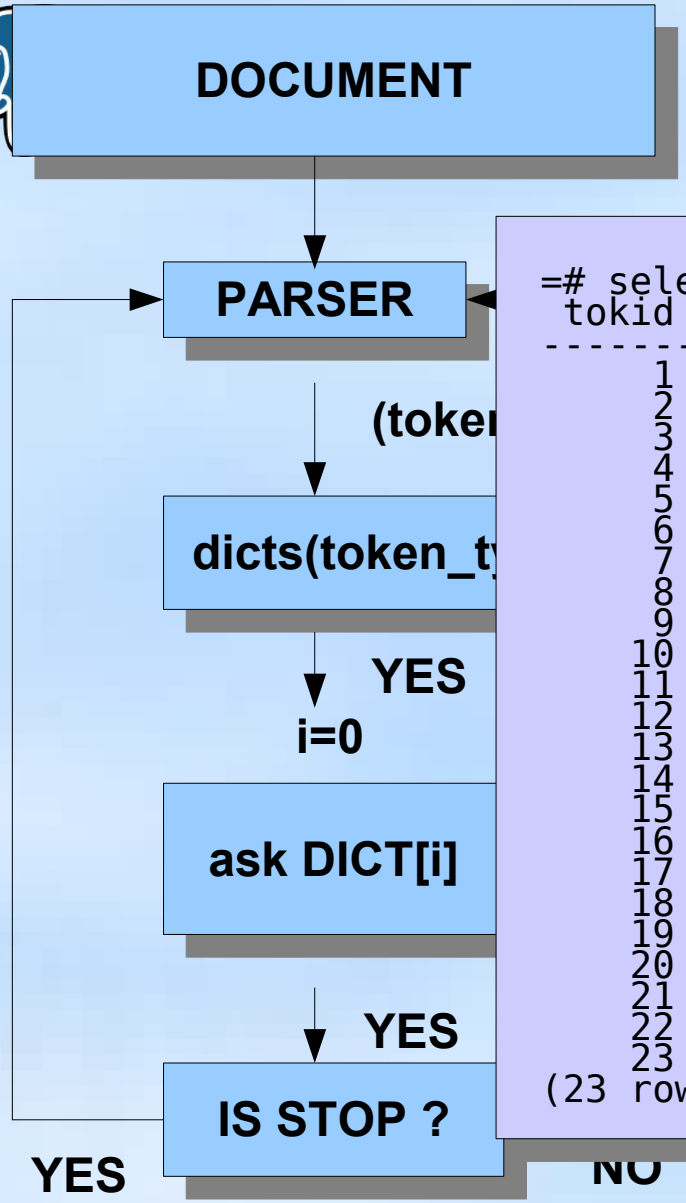
Complete FTS reference

- Data types
  - tsvector, tsquery
- FTS operator
  - @@
- Basic functions
  - to_tsvector, setweight, to_tsquery, plainto_tsquery, ts_rewrite, tsvector_update_trigger
- Additional functions
  - ts_rank_cd, ts_rank, ts_headline
- Additional operators
  - @>, <@
- Debug functions
  - ts_lexize, ts_debug, ts_parse, ts_token_type, numnode, querytree, ts_stat

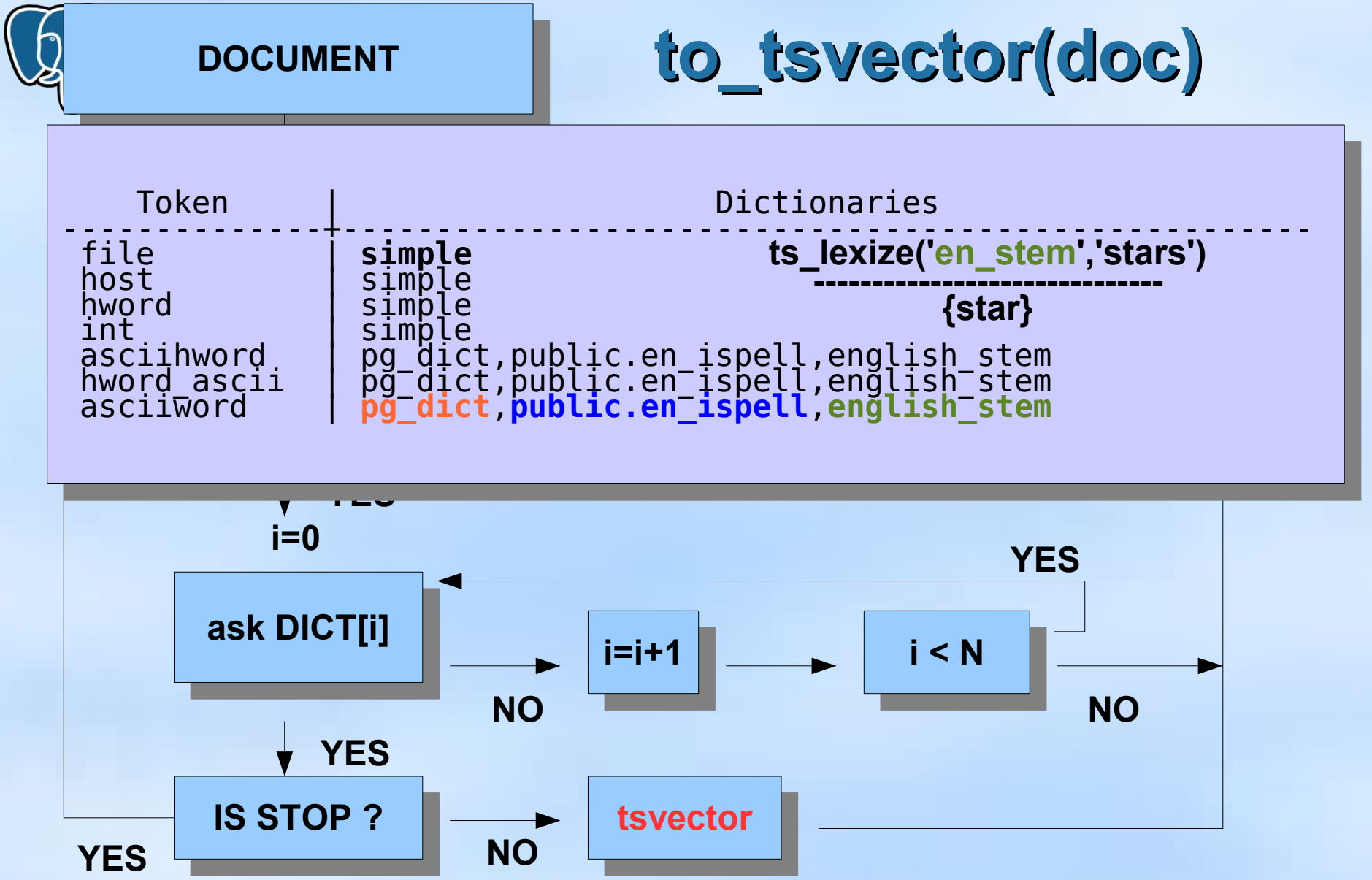
to_tsvector(doc)



to_tsvector(doc)




```
=# select * from ts_token_type('default');
 tokid |      alias      |                description
-------+-----------------+-------------------------------------
     1 | asciiword       | Word, all ASCII
     2 | word            | Word, all letters
     3 | numword         | Word, letters and digits
     4 | email           | Email address
     5 | url             | URL
     6 | host            | Host
     7 | sfloat          | Scientific notation
     8 | version         | Version number
     9 | hword_numpart   | Hyphenated word part, lette
    10 | hword_part      | Hyphenated word part, all
    11 | hword_asciipart | Hyphenated word part, all ASCI
    12 | blank           | Space symbols
    13 | tag             | XML tag
    14 | protocol        | Protocol head
    15 | numhword        | Hyphenated word, letters digit
    16 | asciihword      | Hyphenated word, all ASCII
    17 | hword           | Hyphenated word, all letters
    18 | url_path        | URL path
    19 | file            | File or path name
    20 | float           | Decimal notation
    21 | int             | Signed integer
    22 | uint            | Unsigned integer
    23 | entity          | XML entity
(23 rows)
```


DOCUMENT
to_tsvector(doc)
Token | Dictionaries
file | simple
host | simple
hword | simple
int | simple
asciihword | pg_dict,public.en_ispell,english_stem
hword_ascii | pg_dict,public.en_ispell,english_stem
asciiword | pg_dict,public.en_ispell,english_stem
ts_lexize('en_stem','stars')
{star}
i=0
ask DICT[i]
YES
NO
i=i+1
i < N
NO
YES
IS STOP ?
YES
NO
tsvector

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
-

```
select ts_lexize('intdict', 11234567890);
  ts_lexize
----------
 {112345}
```

Dictionaries

- Dictionary for roman numerals

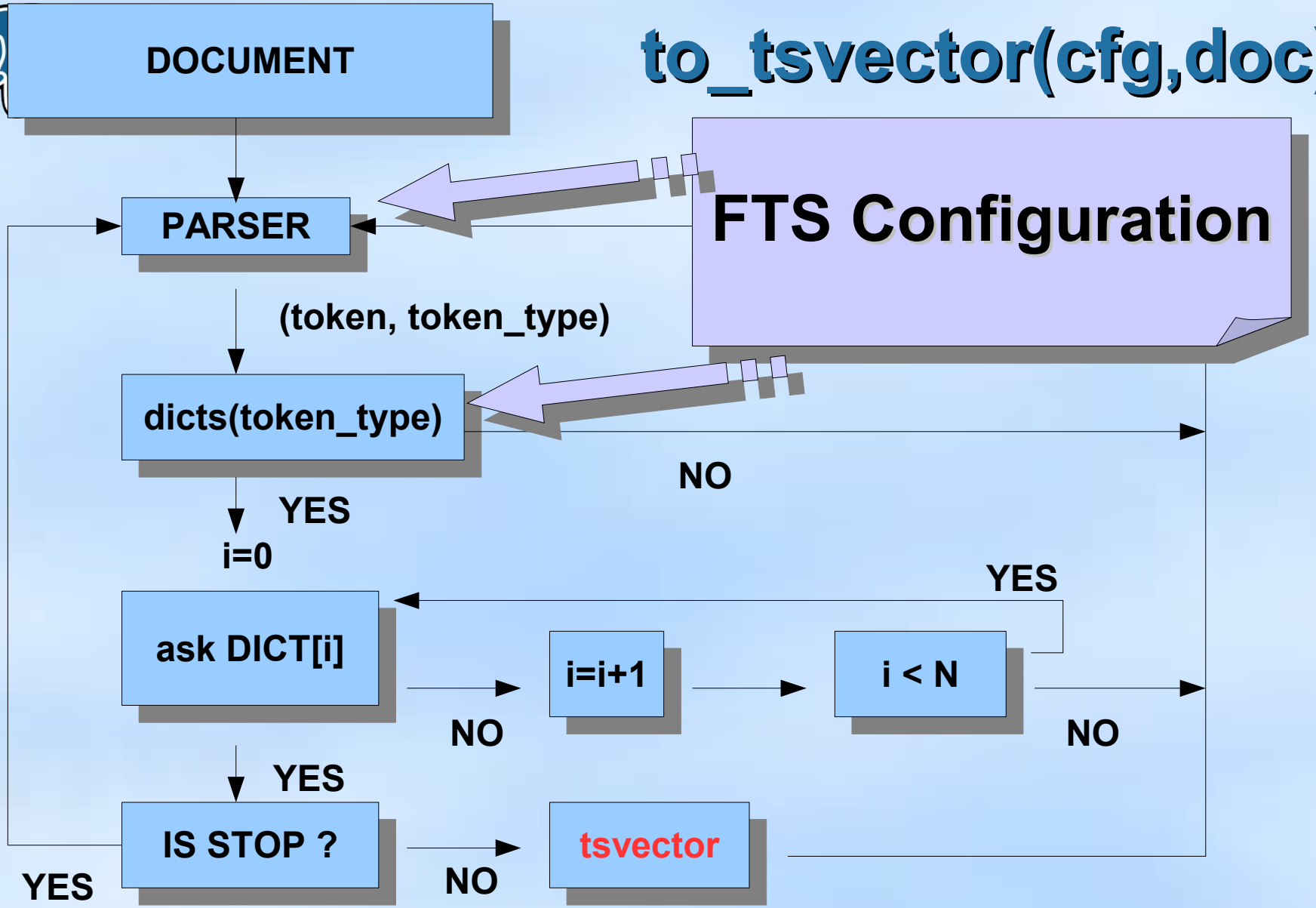
```
=# select ts_lexize('roman', 'XIX');
 ts_lexize
--------
 {19}

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

Dictionaries

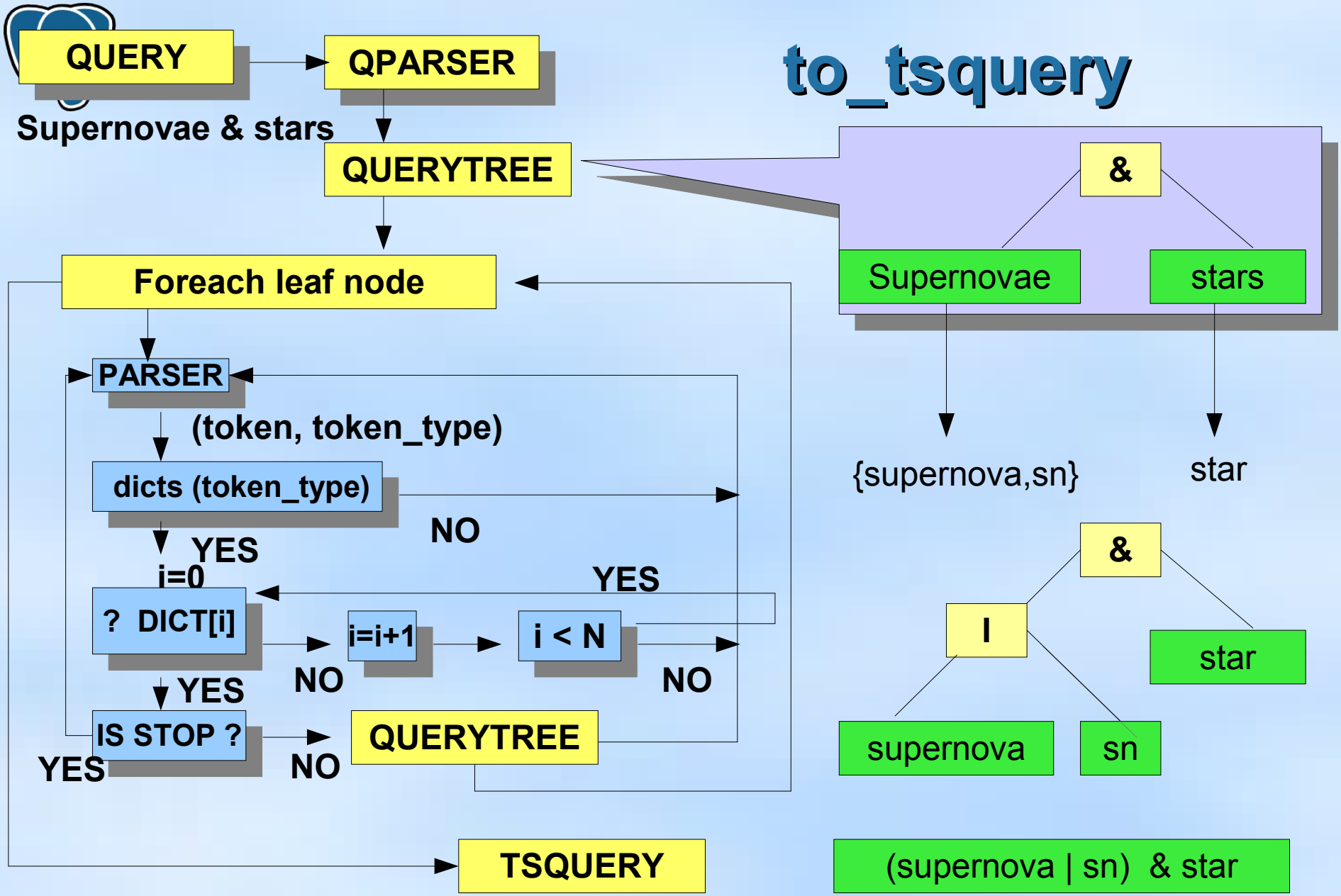
- Dictionary with regexp support (pcre library)

```
# 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
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

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

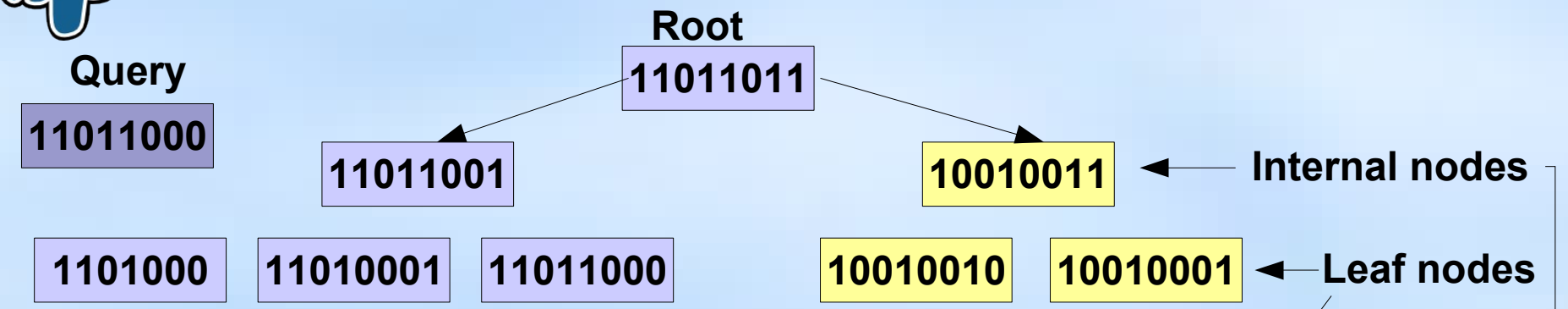
Document is a bitwise-OR of all signatures

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)
```

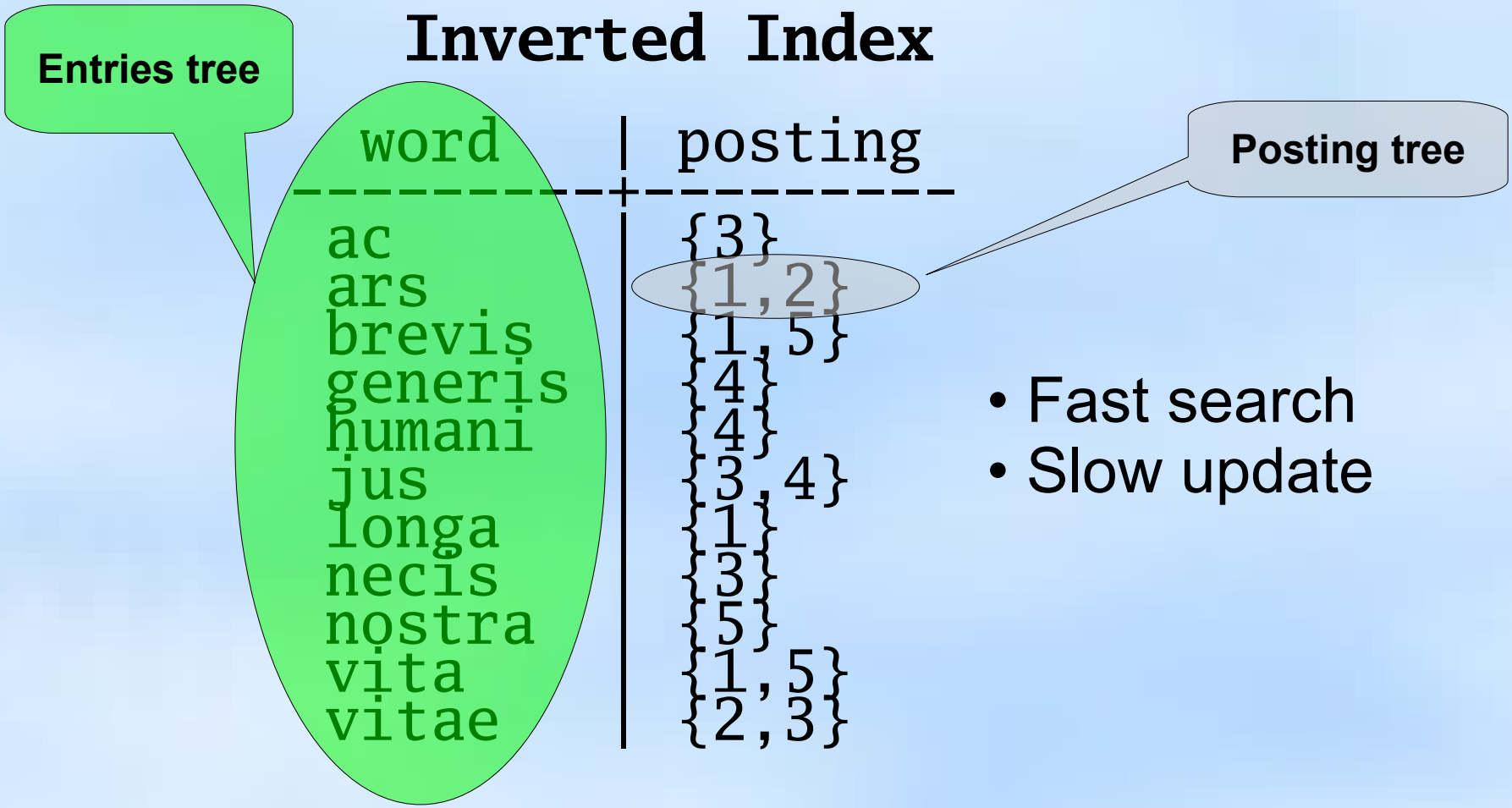
GIN Index

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



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

GiN or GiST ? UPDATE: 8.4 !

- Fastupdate GIN !
  - CREATE INDEX gin_idx ON tt2 USING gin (array_int) WITH (**fastupdate**=1);
  - GIN update speed is comparable with 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/tsearch_data)

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 default_text_search_config;
 default_text_search_config
----------------------------
 pg_catalog.russian
```

APOD example

```
postgres=# \dF+ pg_catalog.russian
Configuration "pg_catalog.russian"
Parser name: "pg_catalog.default"

      Token       | Dictionaries
------------------+--------------
 asciihword       | english_stem
 asciiword        | english_stem
 email            | simple
 file             | simple
 float            | simple
 host             | simple
 hword            | russian_stem
 hword_asciipart  | english_stem
 hword_numpart    | simple
 hword_part       | russian_stem
 int              | simple
 numhword         | simple
 numword          | simple
 sfloat           | simple
 uint             | simple
 url              | simple
 url_path         | simple
 version          | simple
 word             | russian_stem
```

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,ts_rank_cd(fts, q) from apod,
to_tsquery('supernovae & x-ray') q
where fts  @@ q order by ts_rank_cd desc limit 5;
                        title                         | ts_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
```

ts_rank_cd uses only local information !

0 < rank/(rank+1) < 1

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

APOD example: headline

```
postgres=# select ts_headline(body,q,'StartSel=<,StopSel=>,MaxWords=10,MinWords=5'),
ts_rank_cd(fts, q) from apod, to_tsquery('supernovae & x-ray') q where fts  @@
q order by ts_rank_cd desc limit 5;
                              ts_headline                                | ts_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,ts_rank_cd(fts, q) from apod,
to_tsquery('supernovae:b & x-ray') q
where fts  @@ q order by ts_rank_cd desc limit 5;
                    title                     | ts_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

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

790 times

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

Time: 723.634 ms
```

10 times !

```
select id,ts_headline(body,q),rank from (
  select id,body,q, ts_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 ts_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 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 asciihword,asciiword,hword_asciipart 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

ts_rewrite (tsquery, tsquery, tsquery)

ts_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 ts_rewrite(to_tsquery('supernovae'),
'select * from aliases');
      ts_rewrite
--------------------
 'supernova' | 'sn'
```

FTS tips – Query rewriting

```
apod=# select title, ts_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, ts_rank_cd(fts,q,1) as rank
 from apod, ts_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, ts_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;
```

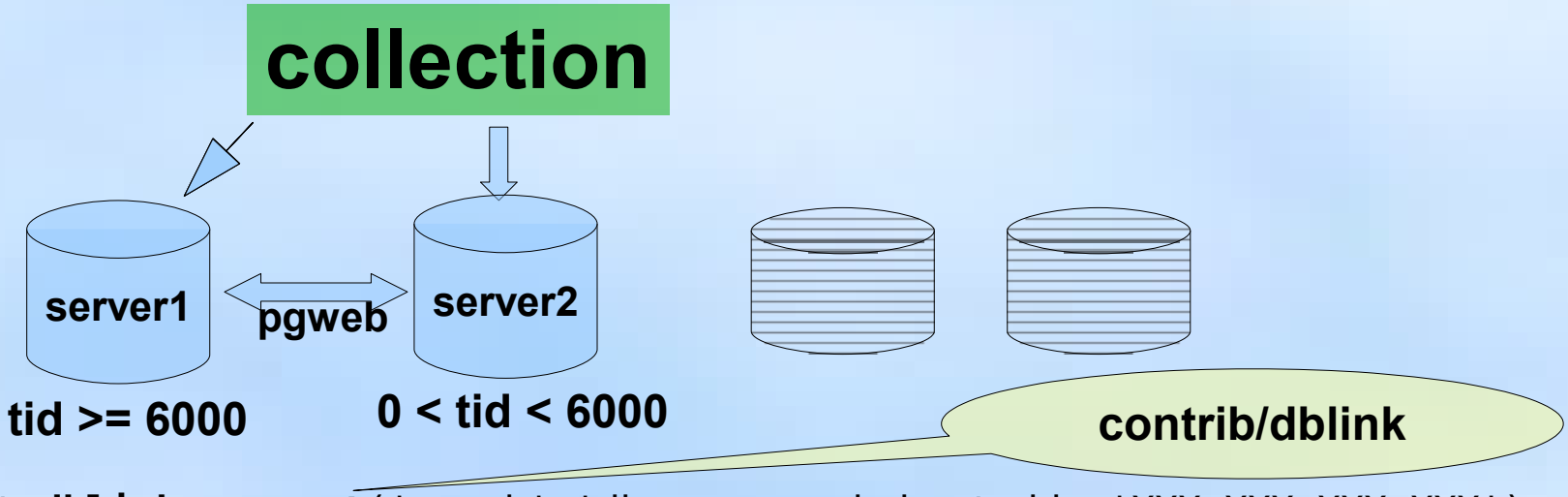
FTS tips – Partition your data

- 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




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

select * from dblink('pgweb',
 'select  tid, title, ts_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,  ts_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.postgresql.org/docs/current/static/textsearch.html
  - http://www.sai.msu.su/~megera/wiki/tsearch2 - tsearch2 Wiki
- **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

FTS 8.4

- PostgreSQL 8.3 has problem with nepali

```
select * from ts_parse('default','मदन पुरस्कार पुस्तकालय');
2     मदन
12
2     पुरस
12
2     कोर
12
2     पुस
12
2     तकोलय
```

FTS 8.4

मदन पुरस्कार पुस्तकालय

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	र्	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

FTS 8.4

- Devanagari script support (better Unicode support)

```
postgres=# set client_encoding to UTF8;
postgres=# 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

FTS 8.4

- Filtering dictionaries support !
  - Recognize, process and pass lexeme to the next dictionary

```
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','Hôtels de la Mer');
    to_tsvector
--------------------
 'hotel':1 'mer':4
(1 row)

'Hôtels'-> 'Hotels' -> 'hotel'
unaccent   french_stem


=# select to_tsvector('fr','Hôtel de la Mer') @@ to_tsquery('fr','Hotels');
 ?column?
----------
 t
(1 row)
=# select ts_headline('fr','Hôtel de la Mer',to_tsquery('fr','Hotels'));
      ts_headline
-------------------------
  <b>Hôtel</b> de la Mer
(1 row)
```

FTS 8.4

- Prefix search support
  - to_tsquery('supernov:*') will match all words with prefix 'supernov'. No overhead, just using new GIN partial match feauture
  - Good if there is no stemmer available

FTS 8.4

- Prefix search support — dictionaries

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

=# select to_tsvector('tst','indices');
 to_tsvector
-------------
 'index':1
(1 row)
```