Experimenting with Phrase Text Search and New RUM Index

Galvanize
November 12, 2016
Austin, Texas, USA

John Scott
CTO of RJ2 Technologies

"Facts", Immutablility and Blobs

- a fact is always in the past tense
- a sha digest is the key to the pdf
- the pdf blob does not need to be stored in postgres

What is Full Text Search?

- Documents are plain old text, in database encoding
- Supports Many Text Encoding
- UTF-8 Encoding of Text Simplifies
- Only Two Datatypes Implement Text Search

Who Wrote Full Text Search?

- Postgres Pro, a Russian Company
- Oleg Bartunov
- Teodor Sigaev
- Artur Zakirov

Think of Full Text Search as a Toolkit to Build Text Search Engines

- relational model excellent for correlating text with analytic data
- chemical equations
- gene sequences
- text fields in spreadsheet
- elastic search still better for text only search

Source of the PDF Documents for the Demo

- arvix.org (Cornell, won the MacArthur Genius Award)
- Theory of Computing Blog Aggregator (http://feedworld.net/toc/)
- Electronic Colloquim on Computational Complexity eccc.hpi-web.de
- Many academic web sites surfed for past 10 years.

Statistics of PDF Documents for Today

PDF Count	154,592 PDF Docs
PDF Page Count	5,526,409 Pages
PDF Total Byte Count	148 GB
Average #Pages per PDF	36 Pages
Text (Lexeme) Search Vector Table Size	19 GB
Text Table Size	8.5 GB

Example of "Google" Style Search

```
NEURAL CRYPTOGRAPHY
```

Full Text Search Data Types

tsvector

```
sorted list of lexemes (words) in the text, with optional positions for proximity ranking
```

tsquery

```
boolean combinations of lexemes and the "followed by" <-> operator for phrases
```

TSVector Data Type

```
SELECT 'a fat cat sat on a mat and ate a fat rat'::tsvector;
                      tsvector
----------------------------------------------------
 'a' 'and' 'ate' 'cat' 'fat' 'mat' 'on' 'rat' 'sat'
```

Positions of Lexemes Needed for Proximity Ranking

```
SELECT to_tsvector('english', 'The Fat Rats');
   to_tsvector
-----------------
 'fat':2 'rat':3
```

Maps (Stem) Words to Lexemes and Removes Stop Words

```
SELECT to_tsvector('english', 'The Suffix Trees');
     to_tsvector
---------------------
 'suffix':2 'tree':3
```

TSQuery Data Type

Boolean AND, OR, NOT with Grouping of Lexemes

```
SELECT 'fat & (rat | cat)'::tsquery;
          tsquery
---------------------------
 'fat' & ( 'rat' | 'cat' )
```

Rearranges nested operators into a logically equivalent formulation

```
SELECT '(fat | rat) <-> cat & !mouse'::tsquery;
                     tsquery
-------------------------------------------------
 ( 'fat' <-> 'cat' | 'rat' <-> 'cat' ) & !'mouse'
```

<-> is "Followed By" Operator.

<6> Means Exactly 6 Words Apart

TSQuery Data Type

Lexemes in a tsquery can be labeled with * to specify prefix matching

```
SELECT 'super:*'::tsquery;
  tsquery
-----------
 'super':*
```

Match any word in a tsvector that begins with "super"

How to Query Text with the Boolean @@ Operator

TRUE

```
SELECT to_tsvector('fat cat') @@ to_tsquery('(fat | rat) <-> cat');
?column?
----------
 t
```

FALSE

```
SELECT to_tsvector('fat rat') @@ to_tsquery('(fat | rat) <-> cat');
 ?column?
----------
 f
```

Extracting Matching Snippets for a "Headline"

```
SELECT ts_headline('now is the time for all good men', 'good'::tsquery);
               ts_headline
-----------------------------------------
 now is the time for all <b>good</b> men
```

Only Highlights Keywords (for Now)

Not Smart About Logical Structure of the Query (yet)

Very, Very Expensive, So Be Sure Push to Target List!

Text Search Indexes - Speed Up the Query

GIN - Typical Inverted Index - Production Many Years

```
CREATE INDEX pdf_page_idx ON pdf_page USING GIN (to_tsvector('english', doc));
```

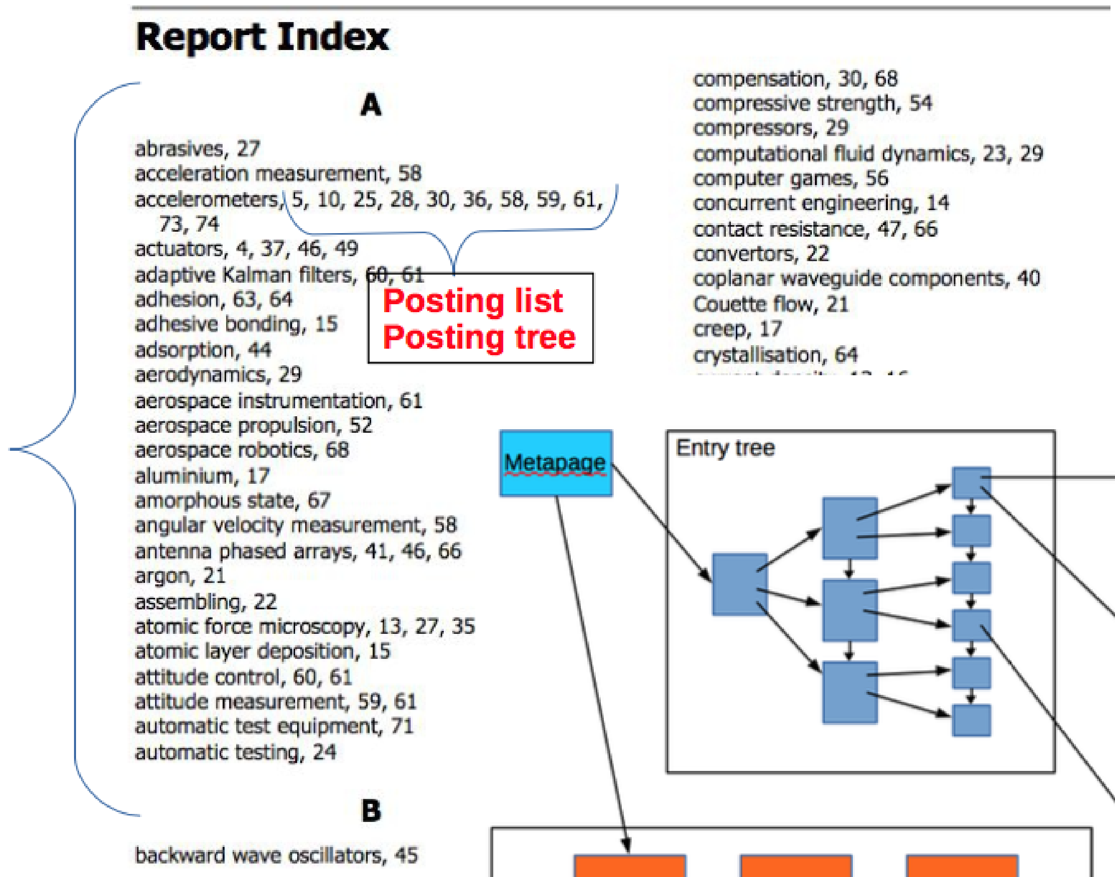
RUM - Still in Beta

```
CREATE INDEX pdf_page_idx ON pdf_page USING GIN (to_tsvector('english', doc));
```

Postgres PROFESSIONAL

No positions in index !

Inverted Index in PostgreSQL



E N T R Y T R E E

Report Index

A

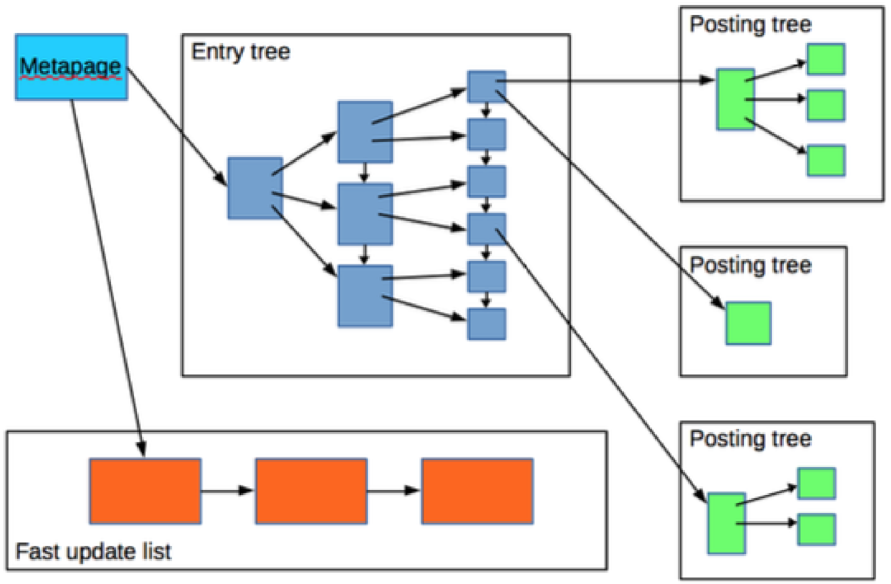
abrasives, 27
acceleration measurement, 58
accelerometers, 5, 10, 25, 28, 30, 36, 58, 59, 61,
73, 74
actuators, 4, 37, 46, 49
adaptive Kalman filters, 60, 61
adhesion, 63, 64
adhesive bonding, 15
adsorption, 44
aerodynamics, 29
aerospace instrumentation, 61
aerospace propulsion, 52
aerospace robotics,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

B

backward wave oscillators, 45

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

Posting list
Posting tree



RUM Index, Still in Beta

- Lexeme Positions Stored in the Index
- Rank Matches with Query Distance Operator that Understands Logical Queries
- Allows Timestamps Materialized in the Index
- Much Smarter Optimzer to Exploit Phrase Structure
- Query Fragments can be Indexed!

RUM Index for Document Classification!

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

BOOM

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

Dictionary and Text Configuration

- Map "Words" onto Lexemes - Stemming
- Can Search Multiple Dictionary in Particular Order
- "simple" is exact word, usually final disction in the search
- Loaded as Extension into Shared Memory (9.6)
- Common Mispellings can be in a Dictionary

Thank you

Galvanize
November 12, 2016
Austin, Texas, USA

John Scott
CTO of RJ2 Technologies

jmscott@rj2tech.com (mailto:jmscott@rj2tech.com)
https://github.com/jmscott/talk (https://github.com/jmscott/talk)