Text Search in 8.4?+

- WIP: Phrase Search
  - Algebra for text queries
- Fast approximate statistics based on GIN index
- Prefix search support (GIN partial match)
- Middleware for text search configurations

Oleg Bartunov, Teodor Sigaev, Mikhail Prokhorov
Moscow University, SAI, Russia
What is a phrase?

- It's tsquery
  - 'a b c'::tsquery
- Ordering is important
  - 'a b c' != 'a c b'
- Distance between words is important
  - 'a b x c' != 'a b c'
Phrase Search

- **What is a phrase?**
  - It's `tsquery`
    - `'a b c'::tsquery`
  - Ordering is important
    - `'a b c' != 'a c b'`
  - Distance between words is important
    - `'a b x c' != 'a b c'`
Phrase Search

Why there is no phrase search support?
- There are already support for boolean operations.
- There is positional information for each lexeme.
Motivation for Algebra

- Existing operators defined at document level
  - 'A & B': tsquery means intersections of two sets

Documents with 'A'

Documents with 'B'
Motivation for Algebra

- Operators AND, OR, AND NOT work with sets

\[ A \text{ OR } B \]
\[ A \text{ AND } B \]
\[ A \text{ NOT } B \]
Motivation for Algebra

- Phrase Search requires operation at *lexeme* level — operator BEFORE ($)
- Different semantics - A NOT B
  - Normal search: Document with A and at the same time without B
  - Phrase search: «A $ X» (X is anything, except B)
Motivation for Algebra

- Phrase can be very complex
  - Even simplest phrase can be transformed to a complex expression.

  \[
  \text{to\_tsquery('nb', 'telefonsvarer')} \Rightarrow
  \text{'telefonsvarer' | 'telefon' & 'svar'}
  \]

  \[
  \text{to\_tsquery('footballklubber $ SMTH')} \Rightarrow
  \text{'( (football & klubber) | (foot & ball & klubber) ) $ SMTH'} \Rightarrow
  \text{hard, but it's not the hardest case}
  \]
Motivation for Algebra

- Phrase can be constructed by a program, or manually using casting (SMTH::tsquery)

«A $( B $ !(C $ !D))»

- We need well-defined algebra for operations: & | ! $

- Backward compatibility!
Motivation for Algebra

- We introduced «generalized» phrase

\[ a \, \left[ \begin{array}{c} n \end{array} \right] \, b \]

- Operator BEFORE (\( \left[ \begin{array}{c} n \end{array} \right] \)) guarantees
  - An order of operands — a BEFORE b
  - Distance between operands, default is 1

\[ a \, \left[ \begin{array}{c} n \end{array} \right] \, b \, \equiv \, a \, \& \, b \, \& \, (\exists \, i,j : \text{pos}(b)_i - \text{pos}(a)_j = n) \]

\[ a \, \$ \, b \, \equiv \, a \, \& \, b \, \& \, (\exists \, i,j : \text{pos}(b)_i - \text{pos}(a)_j = 1) \]
Operations

- $a \ [n] b = b \ [-n] a$
- !(a \ [n] b) = !a \ | \ !b \ | (\ \forall \ i,j: \text{pos}(b) - \text{pos}(a) \neq n)$
- !!(a \ [n] b) = a \ [n] b$
- $a \ !b = a \ & (\ \exists \ i,j: \text{pos}(b) \neq 1)$
- !a \ $ b = b \ & (\ \exists \ i,j: \text{pos}(b) \neq 1)$
- !a \ $ !b = (\ \exists \ i,j: \text{pos}(b) \neq 1)$
- $a \ (b \ | \ c) = a \ b \ | \ a \ c$
- $(b \ | \ c) \ a = b \ a \ | \ c \ a$
- $a \ (b \ & \ c) = b \ & \ c \ & (a \ b \ | \ a \ c)$
- $(b \ & \ c) \ a = b \ & \ c \ & (b \ a \ | \ c \ a)$
Recursive definition

\[(a \ [n] b) \ [m] c = (a \ [n] b) \ & \ c \ & \ (\exists i,j: posL(c)_j - posR(“ab”)_i = m) \implies (a \ [n] b) \ & \ c \ & \ (\exists i,j: pos(c)_j - posR(“ab”)_i = m) \implies (a \ & \ b \ & \ (\exists k,l: pos(b)_k - pos(a)_l = n)) \ & \ c \ & \ (\exists i,j: pos(c)_j - posR(“ab”)_i = m) =
\]

\[= a \ & \ b \ & \ c \ & \ (\exists k,l: pos(b)_k - pos(a)_l = n) \ & \ (\exists j: pos(c)_j - pos(b)_k = m) =
\]

\[= a \ & \ b \ & \ c \ & \ (\exists j,k,l: pos(b)_k - pos(a)_l = n \ & \ pos(c)_j - pos(b)_k = m)
\]

\[= a \ [n] b \ [m] c
\]

\[a \ [n] (b \ [m] c) = a \ [n] b \ [m] c \ (\text{as above})
\]
Query: "black" $ ("hole" | "nebulae") ==> 
"black" $ "hole" | "black" $ "nebulae"

∃ pos("hole")-pos("black")=1
∃ pos("hole")-pos("nebulae")=1
Query: «close» $ «galaxies»

After dictionary: «close» $ («m33» |
(«andromeda» $ «nebulae» | («magellanic» & «clouds»))

Phrase: «close» $ «m33» |
( «close» $ («andromeda» $ «nebulae» )) |
( «magellanic» & «clouds» &
( «close» $ «magellanic» | «close» $ «clouds»))
Phrase Search

- Possible extensions
  - $\#[n]$ — soft $\$$[n]$, order doesn't important
  - $a<$$[n]$ b — at most n words between operands
  - $a $$[n]$$> b$ — at least n words between operands
  - And so on ...
Partial Match for GIN

- Prefix search for a text search
- Improve performance `LIKE '%%foo%%'`
  - It's not a full text search
  - Btree index (text_pattern_ops) can improve
    - `LIKE '%%FOO'`
    - `LIKE 'FOO%%'`
Index all permutations of string!

`=# select permute('hello');
   permute

{hello$, ello$h, llo$he, lo$hel, o$hell}

'$$' is used for visualization, we use \0

LIKE ' %l% ' =>  ~ ' l* ' 
LIKE ' h%o ' =>  ~ ' o$h* ' 
LIKE ' %o ' =>  ~ ' o$* ' 
LIKE ' h% ' =>  ~ ' h*$ `
### Partial Match: wildspeed

750,000 words, average length is 8 characters, time in ms

<table>
<thead>
<tr>
<th></th>
<th>h%</th>
<th>hel%</th>
<th>h%o</th>
<th>%l%</th>
<th>%lll%</th>
<th>%l</th>
<th>%lll</th>
<th>%ll%o</th>
</tr>
</thead>
<tbody>
<tr>
<td>wildspeed</td>
<td>28.0</td>
<td>1.1</td>
<td>1.1</td>
<td>434</td>
<td>0.7</td>
<td>426</td>
<td>0.7</td>
<td>18</td>
</tr>
<tr>
<td>Btree/seqscan</td>
<td>8.5</td>
<td>1.0</td>
<td>8.6</td>
<td>415</td>
<td>408</td>
<td>407</td>
<td>404.0</td>
<td>404</td>
</tr>
</tbody>
</table>

CREATE INDEX ... USING btree (w text_pattern_ops) : 3.175 seconds

CREATE INDEX ... USING gin (w2 wildcard_ops) : 1 hour 10 minutes
Prefix search

The popular request for the text search

```sql
SELECT 'superstar on party'::tsvector @@ 'super:*' AS yes;
   yes
-----
t

SELECT 'supernovae:1A sky:2B'::tsvector @@ 'super:A*' AS yes;
   yes
-----
t
```
Prefix Search

- Based on partial match algorithm in GIN
- Syntax — use flag '*'
  - 'abc:*':tsquery - search documents with words 'abc*'
- Prefix search comes for free, no special actions required!
- Dictionary API supports prefix flag
Prefix search

- `tsquery @@ to_tsquery('supernova:a* & stars')`
  - Find **supernova** in **titles**

```sql
=# select count(*) from papers where fts @@
  to_tsquery('supernova:a* & stars');

<table>
<thead>
<tr>
<th>count</th>
</tr>
</thead>
<tbody>
<tr>
<td>838</td>
</tr>
</tbody>
</table>

(1 row)
```

```sql
=# select count(*) from papers where fts @@
  to_tsquery('supernova:a & stars');

<table>
<thead>
<tr>
<th>count</th>
</tr>
</thead>
<tbody>
<tr>
<td>835</td>
</tr>
</tbody>
</table>

(1 row)
```
Fast approximate statistics

- **Gevel extension** — GiST/GIN indexes explorer
  (http://www.sai.msu.su/~megera/wiki/Gevel)

- **Fast** — uses only GIN index (no table access)

- **Approximate** — no table access, which contains visibility information, approx. for long posting lists

- Statistics looks good for mostly **read-only** data
Fast approximate statistics

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

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

<table>
<thead>
<tr>
<th>word</th>
<th>ndoc</th>
</tr>
</thead>
<tbody>
<tr>
<td>page</td>
<td>340858</td>
</tr>
<tr>
<td>figur</td>
<td>240366</td>
</tr>
<tr>
<td>use</td>
<td>148022</td>
</tr>
<tr>
<td>model</td>
<td>134442</td>
</tr>
<tr>
<td>result</td>
<td>129010</td>
</tr>
</tbody>
</table>

(5 rows)

Time: 520.714 ms
Fast approximate statistics

- gin_stat() vs ts_stat()

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

...wait....

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

<table>
<thead>
<tr>
<th>word</th>
<th>exact</th>
<th>estimation</th>
<th>error</th>
</tr>
</thead>
<tbody>
<tr>
<td>page</td>
<td>340430</td>
<td>340858</td>
<td>0.13%</td>
</tr>
<tr>
<td>figur</td>
<td>240104</td>
<td>240366</td>
<td>0.11%</td>
</tr>
<tr>
<td>use</td>
<td>147132</td>
<td>148022</td>
<td>0.60%</td>
</tr>
<tr>
<td>model</td>
<td>133444</td>
<td>134442</td>
<td>0.74%</td>
</tr>
<tr>
<td>result</td>
<td>128977</td>
<td>129010</td>
<td>0.03%</td>
</tr>
</tbody>
</table>

(5 rows)

Time: 550.562 ms
Middleware for FTS configuration

- Dictionaries should be able to specify how interpret their output. For example, dictionary returns (a,b). Possible interpretations:
  - (a,b) -> a & b
  - (a,b) -> a | b
  - (a,b) -> a $[n]$ b - 'b' follows 'a', n words max
  - (a,b) -> a #[n] b - soft $ (no order)$
  - Etc.
Middleware for FTS configuration

- Option for dictionary to return also an original word
- Manage how word is processed by a stack of dictionaries
  - Stop if recognized — current behaviour
  - Process and continue — filters
    - Use case: accent removal problem (wrong highlighting) — cannot be solved by using function `to_tsvector(remove_accent(document))`
Text Search in 8.4?+

- This work is supported by
  - jfg://networks — over-blog.net
  - EnterpriseDB

Oleg Bartunov, Teodor Sigaev, Mikhail Prokhorov
Moscow University, SAI, Russia