pg_similarity
functions and operators for executing similarity queries

Euler Taveira de Oliveira

4Linux
PostgreSQL Brasil

21 de maio de 2009
1. Introduction
2. PostgreSQL
3. pg_similarity
Approximate Queries

- database works with a **exact** query model
  - key $\approx$ value
Approximate Queries

- database works with a **exact** query model
  - key = value
- we want to be able to do some **almost** exact queries
  - key ≈ value
Approximate Queries

- database works with a **exact** query model
  - key $\equiv$ value
- we want to be able to do some **almost** exact queries
  - key $\approx$ value
- and to be able to define how flexible it is
  - similarity: $\phi > \text{threshold}$
Problems

- human
  - mistyping
  - imprecise typing (abbreviation, omission, ...)
  - insufficient information
Problems

- human
  - mistyping
  - imprecise typing (abbreviation, omission, ...)
  - insufficient information

- application
  - application with problems (truncation, abbreviation – without user asks for)
  - data model failed
    - inexistent or imprecise constraints
    - inexistent foreign keys
Problems

- **human**
  - mistyping
  - imprecise typing (abbreviation, omission, ...)
  - insufficient information

- **application**
  - application with problems (truncation, abbreviation – without user asks for)
  - data model failed
    - inexistent or imprecise constraints
    - inexistent foreign keys

- **obsolescence**
  - real world is dynamic!
Consequences

- poor data quality
  - "2% of records in a customer file become obsolete in one month because customers die, divorce, marry and move [DWI02]
  - Bill Clinton, William Jefferson Clinton, and William Jefferson Blythe III: same person?
Consequences

- poor data quality
  - "2% of records in a customer file become obsolete in one month because customers die, divorce, marry and move" [DWI02]
  - Bill Clinton, William Jefferson Clinton, and William Jefferson Blythe III: same person?

- high financial impact
  - "poor data quality cost USA businesses a staggering USD 611 bi/year in postage, printing and staff overhead" [DWI02]
  - "Wrong price data in retail databases costs American consumers USD 2.5 billion in annual overcharges" [E00]
Solution?

Approximate Queries

Performance

Quality
Agenda

1. Introduction
2. PostgreSQL
3. pg_similarity
What is available?

- regular expression
- fuzzystrmatch
- text search
Problems?

- inversion (euler taveira × taveira euler)
- mistyping (euler × euller)
- stopwords (euler de oliveira × euler oliveira)
- stemming (similarity × similarities)
- flexibility (change tokenizer, threshold, etc)
Motivation

- Ideas
  - low response time
  - do not use pre-processing step (online technique)
  - do not use auxiliary structure (catalog or auxiliary table)
  - maintain result quality
  - extensible
Motivation

- **Ideas**
  - low response time
  - do not use pre-processing step (online technique)
  - do not use auxiliary structure (catalog or auxiliary table)
  - maintain result quality
  - extensible

- **Design**
  - similarity functions
  - operators
  - auxiliary functions
Installation

$ cd pg_similarity
$ PATH=/pg/bin:$PATH USE_PGXS=1 gmake install
$ /pg/bin/psql -f /pg/share/contrib/pg_similarity.sql mydb

CREATE FUNCTION
CREATE FUNCTION
CREATE FUNCTION
CREATE FUNCTION

CREATE OPERATOR
CREATE OPERATOR
Similarity Functions

- is a function that calculates how similar are two data
- domain-dependent functions (almost all of them)
- some functions need parsing step (tokenization: space, non-alphanumerics, and n-gram)
- examples

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Cosine</td>
<td>Euclidean</td>
<td>Block</td>
</tr>
<tr>
<td>Jaro-Winkler</td>
<td>Q-Gram</td>
<td>Smith-Waterman</td>
</tr>
<tr>
<td>Dice</td>
<td>Monge-Elkan</td>
<td>Jaccard</td>
</tr>
<tr>
<td>Hamming</td>
<td>Levenshtein</td>
<td>Needleman-Wunsch</td>
</tr>
</tbody>
</table>

Euler Taveira de Oliveira  PGCon 2009
Example: Levenshtein

- \( \min(a[x][y-1] + i, a[x-1][y] + d, a[x-1][y-1] + s) \)
- insertion (i), deletion (d), and substitution (s) cost = 1
it is easier to choose a threshold if you know the limits
almost all of the functions return unnormalized results!

```
euler=# select lev('euler', 'heuser'); -- default
lev
-------
0.666667
(1 row)
```

```
euler=# select lev('euler', 'heuser'); -- hey, that's it!
lev
-----
2
(1 row)
```
**Operators**

- SQL syntax sugar for similarity functions

```sql
euler=# CREATE TABLE foo (a text);
CREATE TABLE

euler=# INSERT INTO foo VALUES('Euler T. de Oliveira'),
euler-# ('Euler Taveira de Oliveira');
INSERT 0 2

euler=# SELECT * FROM foo WHERE a ~@@ 'Euler Taveira';
a
---------------------------
Euler Taveira de Oliveira
Euler T. de Oliveira
(2 rows)
```
euler=# SELECT set_threshold('jarowinkler', 0.9);
set_threshold
--------------
   0.9
(1 registro)

euler=# SELECT * FROM foo WHERE a ~@@ 'Euler Taveira';
a
---------------------------
 Euler Taveira de Oliveira
(1 row)
Auxiliary Functions

- **get_isnormalized(func)**: values are true and false
- **set_isnormalized(func, value)**: switches between normalized and unnormalized results
- **get_threshold(func)**: values are between 0 and 1
- **set_threshold(func, value)**: values greater than value are returned
- **get_tokenizer(func)**: values are spaces, nonalnum, and n-gram
- **set_tokenizer(func, value)**: changes tokenization function for some algorithms
Auxiliary Functions: example

```sql
euler=# select get_threshold('jarowinkler');
   get_threshold
---------------
          0.7
(1 row)

euler=# select set_threshold('jarowinkler', 0.85);
   set_threshold
---------------
          0.85
(1 row)
```
some more similarity functions (listed in TODO)
add some custom guc variables (replace get_* and set_*)
UTF-8 aware?
add some selectivity estimator functions for operators
website
write some docs (the source-code has examples how each function works)
your idea?
http://pgfoundry.org/projects/pgsimilarity/


DWI02 Data Warehousing Institute report 2002.
Questions

Euler Taveira de Oliveira
euler@timbira.com
http://www.timbira.com/