# pg_similarity

## functions and operators for executing similarity queries

Euler Taveira de Oliveira

4Linux
PostgreSQL Brasil

21 de maio de 2009

# Agenda

1 Introduction

2 PostgreSQL

3 pg_similarity

# Approximate Queries

- database works with a **exact** query model
  - key = value

# Approximate Queries

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

# 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
- obsolence
  - 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?

# Agenda

1 Introduction

2 PostgreSQL

3 pg_similarity

# What is available?

- regular expression
- fuzzystrmatch
- text search

# Problems?

- inversion (euler taveira x taveira euler)
- mistyping (euler x eu**ll**er)
- stopwords (euler **de** oliveira x euler oliveira)
- stemming (similarity x similarit**ies**)
- flexibility (change tokenizer, threshold, etc)

# Agenda

1 Introduction

2 PostgreSQL

3 pg_similarity

# 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 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-alphanumeric, and n-gram)
- examples

| Cosine | Euclidean | Block |
|---|---|---|
| Jaro-Winkler | Q-Gram | Smith-Waterman |
| Dice | Monge-Elkan | Jaccard |
| Hamming | Levenshtein | Needleman-Wunsch |

# Example: Levenshtein

| | | **e** | **u** | **l** | **e** | **r** |
|---|---|---|---|---|---|---|
| | 0 | 1 | 2 | 3 | 4 | 5 |
| **h** | 1 | 1 | 2 | 3 | 4 | 5 |
| **e** | 2 | 1 | 2 | 3 | 3 | 4 |
| **u** | 3 | 2 | 1 | 2 | 3 | 4 |
| **s** | 4 | 3 | 2 | 2 | 3 | 4 |
| **e** | 5 | 4 | 3 | 3 | 2 | 3 |
| **r** | 6 | 5 | 4 | 4 | 3 | **2** |

- $min(a[x][y-1]+i, a[x-1][y]+d, a[x-1][y-1]+s)$
- insertion (i), deletion (d), and substitution (s) cost $= 1$

# Normalized x Unnormalized

- 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

```
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)
```

# Operators (cont'd)

```
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

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

euler=# select set_threshold('jarowinkler', 0.85);
 set_threshold
---------------
          0.85
(1 row)
```

# TODO

- 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?

# References

http://pgfoundry.org/projects/pgsimilarity/

E00 English, L. **Information Quality Management: The Next Frontier**. DM Review Magazine, April 2000. http://www.dmreview.com/article_sub.cfm?articleId=2073

DWI02 **Data Warehousing Institute report 2002**.

# Questions

?

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