

pg_similarity

funções e operadores para executar consultas por similaridade

Euler Taveira de Oliveira

Timbira
PostgreSQL Brasil

24 de outubro de 2009

timbira

Agenda

- 1 Introdução
- 2 PostgreSQL
- 3 pg_similarity

timbira

Consultas Aproximadas

- banco de dados trabalha com modelo de consulta **exata**
 - campo = valor

timbira

Consultas Aproximadas

- banco de dados trabalha com modelo de consulta **exata**
 - campo = valor
- nós queremos fazer algumas consultas **quase** exatas
 - chave \approx valor

timbira

Consultas Aproximadas

- banco de dados trabalha com modelo de consulta **exata**
 - campo = valor
- nós queremos fazer algumas consultas **quase** exatas
 - chave \approx valor
- e sermos capazes de definir quão flexível seremos
 - similaridade: $\phi > \text{limiar}$

timbira

Problemas

- humano
 - entrada incorreta
 - informação imprecisa (abreviaturas, omissão, ...)
 - desinformação
- aplicação
 - aplicações com problemas
 - modelo de dados é falho
 - restrições inexistentes ou imprecisas
 - chaves estrangeiras inexistentes
- obsolência
 - mundo real é dinâmico!

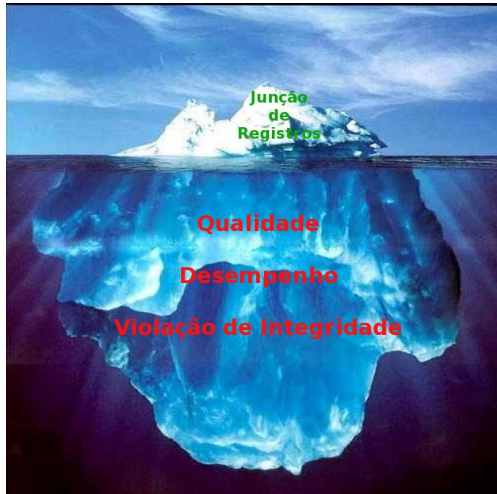
timbira

Consequências

- pobre qualidade dos dados
 - 2% dos registros estão obsoletos em registros de consumidores após 1 mês [DWI02]
 - Bill Clinton, William Jefferson Clinton, William Jefferson Blythe III: mesma pessoa?
- grande impacto financeiro
 - U\$ 611 bi/ano perdido nos USA devido a pobre qualidade dos dados de consumidores [DWI02]
 - U\$ 2,5 bi/ano perdido devido a preços incorretos nos BDs de lojas de varejo [E00]

timbira

Solução?



timbira

Agenda

- 1 Introdução
- 2 PostgreSQL
- 3 pg_similarity

timbira

O que temos?

- expressão regular
- fuzzystrmatch
- busca textual (*text search*)

timbira

Problemas?

- inversão (euler taveira x taveira euler)
- entrada incorreta (euler x euller)
- *stopwords* (euler **de** oliveira x euler oliveira)
- *stemming* (ação x ações)
- flexibilidade (alterar *tokenizer*, limiar, etc)

timbira

Agenda

- 1 Introdução
- 2 PostgreSQL
- 3 pg_similarity

timbira

Motivação

- Idéias
 - **baixo** tempo de resposta
 - **não** utilizar pré-processamento (técnica *online*)
 - **não** utilizar estrutura auxiliar (catálogo ou tabela auxiliar)
 - manter a qualidade dos resultados
 - extensível

timbira

Motivação

- Idéias
 - **baixo** tempo de resposta
 - **não** utilizar pré-processamento (técnica *online*)
 - **não** utilizar estrutura auxiliar (catálogo ou tabela auxiliar)
 - manter a qualidade dos resultados
 - extensível
- Modelagem
 - funções de similaridade
 - operadores
 - funções auxiliares

timbira

Instalação

```
1 $ cd pg_similarity
2 $ PATH=/pg/bin:$PATH USE_PGXS=1 gmake install
3 $ /pg/bin/psql -f /pg/share/contrib/pg_similarity.sql
    mydb
4 CREATE FUNCTION
5 CREATE FUNCTION
6 CREATE FUNCTION
7 .
8 .
9 CREATE OPERATOR
10 .
11 .
```

timbira

Funções de Similaridade

- é uma função que calcula quão similar são dois dados
- funções dependentes do domínio (quase todas elas)
- algumas funções necessitam de um passo de análise (separar em *tokens*: espaço, não-alfanuméricos e n-gramas)
- exemplos

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

timbira

Exemplo: 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

- $\min(a[x][y - 1] + i, a[x - 1][y] + d, a[x - 1][y - 1] + s)$
- custo de inserção (i), deleção (d) e substituição (s) = 1

timbira

Normalizado x Desnormalizado

- é fácil escolher um limiar se você conhece os limites
- **quase** todas as funções retornam resultados desnormalizados

```
1 euler=# select lev('euler', 'heuser'); -- default
2     lev
3 -----
4  0.666667
5 (1 row)
6
7 euler=# select lev('euler', 'heuser'); -- that's it!
8     lev
9 -----
10      2
11 (1 row)
```

timbira

Operadores

- sintaxe SQL para funções de similaridade

```
1 euler=# CREATE TABLE foo (a text);
2 CREATE TABLE
3 euler=# INSERT INTO foo VALUES('Euler T. de Oliveira'),
4 euler=# ('Euler Taveira de Oliveira');
5 INSERT 0 2
6 euler=# SELECT * FROM foo WHERE a ~@@ 'Euler Taveira';
7
8
9 Euler Taveira de Oliveira
10 Euler T. de Oliveira
11 (2 rows)
```

timbira

Operadores (cont)

```
1 euler=# SELECT set_threshold('jarowinkler', 0.9);
2   set_threshold
3  -----
4              0.9
5 (1 registro)
6
7 euler=# SELECT * FROM foo WHERE a ~@@ 'Euler Taveira';
8              a
9  -----
10 Euler Taveira de Oliveira
11 (1 row)
```

timbira

Funções Auxiliares

- **get_isnormalized(func)**: valores são *true* e *false*
- **set_isnormalized(func, valor)**: alterna entre resultados normalizados e desnormalizados
- **get_threshold(func)**: valores estão entre 0 e 1
- **set_threshold(func, valor)**: valores maiores do que *valor* são retornados
- **get_tokenizer(func)**: valores são *spaces*, *nonalnum* e *n-gram*
- **set_tokenizer(func, valor)**: altera função de separação em *tokens* de alguns algoritmos

timbira

Funções Auxiliares: exemplo

```
1 euler=# select get_threshold('jarowinkler');
2   get_threshold
3 -----
4              0.7
5 (1 row)
6
7 euler=# select set_threshold('jarowinkler', 0.85);
8   set_threshold
9 -----
10              0.85
11 (1 row)
```

timbira

TODO

- implementar mais algumas funções de similaridade (listadas no TODO)
- adicionar variáveis personalizadas – GUCs (substituir `get_*` and `set_*`)
- suporte a UTF-8 em algumas funções?
- suporta a indexação em algumas funções?
- adicionar algumas funções para estimar a seletividade para operadores
- website
- escrever alguma documentação (o código tem exemplos de como cada função funciona)
- sua sugestão?

timbira

Referências

<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.**

timbira

Perguntas

?

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

timbira