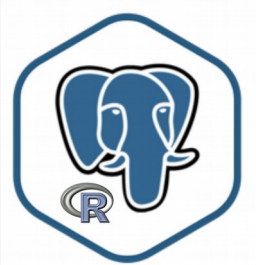
PLR para análisis de logs en PostgreSQL(PG)



Anthony R. Sotolongo León
asotolongo@gmail.com

PgDay Santiago 2019

```
select * from ponente where "usuario"='asotolongo';
```

```
-[ RECORD 1 ]+-------------------------------------------------
usuario      | asotolongo
nombre       | Anthony R. Sotolongo Leon
correo       | asotolongo@gmail.com, anthony.sotolongo@arkadios.cl
trabajo      | ARKADIOS
especialista | BBDD, PostgreSQL, SQL
entusiasta   | Baile, analisis de datos, PLR
libro        | PL/pgSQL y otros lenguajes procedurales en PostgreSQL
social       | https://cl.linkedin.com/in/anthony-sotolongo-93158369,+
             | https://twitter.com/AnthonySotolong
```

Agenda

- Introducción al análisis de logs en PG
- Características PLR en PostgreSQL
- Ejemplos de funciones en PLR
- Caso de uso- Análisis de logs
- Bibliografía
- Trabajos futuros
- Conclusiones

Introducción al análisis de logs en PG

- ¿Qué es un log?
- Es un **registro secuencial/cronológico** en un archivo o BBDD de los **eventos o acciones** que ocurren en servicios/sistemas, que sirve de **evidencia de comportamiento** de donde se generan.

Introducción al análisis de logs en PG

- Antes de analizar un log, ¿Qué hacemos?

Introducción al análisis de logs en PG

- Antes de analizar un log, ¿Qué hacemos?
  - Definir objetivos de análisis, ejemplos:
    - Auditoria
    - Errores
    - Rendimiento
  - Entender el formato que viene:
    - ----------postgresql.conf----------

      # ERROR REPORTING AND LOGGING

      # - When to Log – (log_min_duration_statement, log_min_error_statement ...)

      # - What to Log – (log_connections, log_line_prefix, log_statement ...)

Introducción al análisis de logs en PG

- Antes de analizar un log, ¿Qué hacemos?
  - Configurar!!!
  - Aspectos de análisis, ejemplos:
    - Conexiones
    - Errores
    - DDL.
    - DML.
    - Archivos temporales.
    - etc

Introducción al análisis de logs en PG

- Using CSV-Format Log Output
  - log_destination = 'stderr,csvlog'
  - logging_collector = on

```
SELECT * from  (select logs.files, (pg_stat_file('log/'||logs.files)).modification  from (
SELECT pg_ls_dir as files FROM pg_ls_dir('log')) as logs) as a  order by 2 desc ;
```

files	modification
postgresql-2019-08-21_000000.log	2019-08-21 15:29:14
postgresql-2019-08-21_000000.csv	2019-08-21 15:29:14
postgresql-2019-08-20_094238.log	2019-08-21 11:30:31
postgresql-2019-08-20_094238.csv	2019-08-21 11:30:31
postgresql-2019-08-20_093646.csv	2019-08-20 09:42:38
postgresql-2019-08-20_093646.log	2019-08-20 09:42:38

Introducción al análisis de logs en PG

- Using CSV-Format Log Output

```
CREATE TABLE loga.postgres_log
(
  log_time timestamp(3) with time zone,
  user_name text,
  database_name text,
  process_id integer,
  connection_from text,
  session_id text,
  session_line_num bigint,
  command_tag text,
  session_start_time timestamp with time zone,
  virtual_transaction_id text,
  transaction_id bigint,
  error_severity text,
  sql_state_code text,
  message text,
  detail text,
  hint text,
  internal_query text,
  internal_query_pos integer,
  context text,
  query text,
  query_pos integer,
  location text,
  application_name text,
  PRIMARY KEY (session_id, session_line_num)
);
```

```
copy loga.postgres_log from
'/var/lib/postgresql/10/main/log/archivo.csv' WITH csv;
```

Introducción al análisis de logs en PG

- Using CSV-Format Log Output

```
CREATE EXTENSION file_fdw;

CREATE SERVER filelog FOREIGN DATA WRAPPER file_fdw;
```

```
CREATE FOREIGN TABLE loga.pglog (
  log_time timestamp(3) with time zone,
  user_name text,
  database_name text,
  process_id integer,
  connection_from text,
  session_id text,
  session_line_num bigint,
  command_tag text,
  session_start_time timestamp with time zone,
  virtual_transaction_id text,
  transaction_id bigint,
  error_severity text,
  sql_state_code text,
  message text,
  detail text,
  hint text,
  internal_query text,
  internal_query_pos integer,
  context text,
  query text,
  query_pos integer,
  location text,
  application_name text
) SERVER filelog
OPTIONS ( filename
'/var/lib/postgresql/10/main/log/postgresql-2019-08-
20_094238.csv', format 'csv' );
```

Introducción al análisis de logs en PG

- Se puede consultar por algo específico de los logs

```
select extract (hour from log_time) hora,
extract (minute from log_time) minutos ,
user_name, count(*)
from loga.postgres_log where error_severity='ERROR'
group by 1,2,3
```

hora	minutos	user_name	count
9	55	postgres	1
9	42	postgres	2
9	57	postgres	1
9	56	postgres	1
10	26	postgres	1
9	48	nuevo	34
10	27	postgres	1
10	11	postgres	1
11	14	postgres	1
9	53	postgres	1
9	51	nuevo	9
10	9	postgres	1
10	24	postgres	2
9	43	postgres	1

```
select substring(  connection_from from 1 for position(':' in
connection_from) ) conexion_desde,count(*) from loga.postgres_log
where command_tag='authentication' group by 1
```

conexion_desde	count
127.0.0.1:	9
	3

Introducción al análisis de logs en PG

- Se puede consultar por algo específico de los logs

```
select count(*) from loga.postgres_log where query
ilike '%select%'  and error_severity='ERROR'
```

123 count
50

Introducción al análisis de logs en PG

- ¿Cómo descubrir información en los logs ?

Introducción al análisis de logs en PG

- ¿Como descubrir información en los logs ?

Text Mining

Introducción al análisis de logs en PG

- ¿Como descubrir información en los logs ?

Text Mining

Extensibilidad de PostgreSQL

PostgreSQL soporta que se le agreguen **tipos de datos**, **funciones**, **lenguajes**, etc definidos por el usuario, y se pueden empaquetar mediante extensiones:
CREATE EXTENSION

https://pgxn.org (aprox 290 extensiones)

Características PLR en PostgreSQL

- R: Es un lenguaje estadístico de código abierto, con “cientos” o “miles” de bibliotecas de funcionalidades(comprobadas) de análisis de datos, con excelente documentación y soporte

Características PLR en PostgreSQL

http://pypl.github.io/PYPL.html
PopularitY of Programming Language Index

Worldwide, Oct 2019 compared to a year ago:

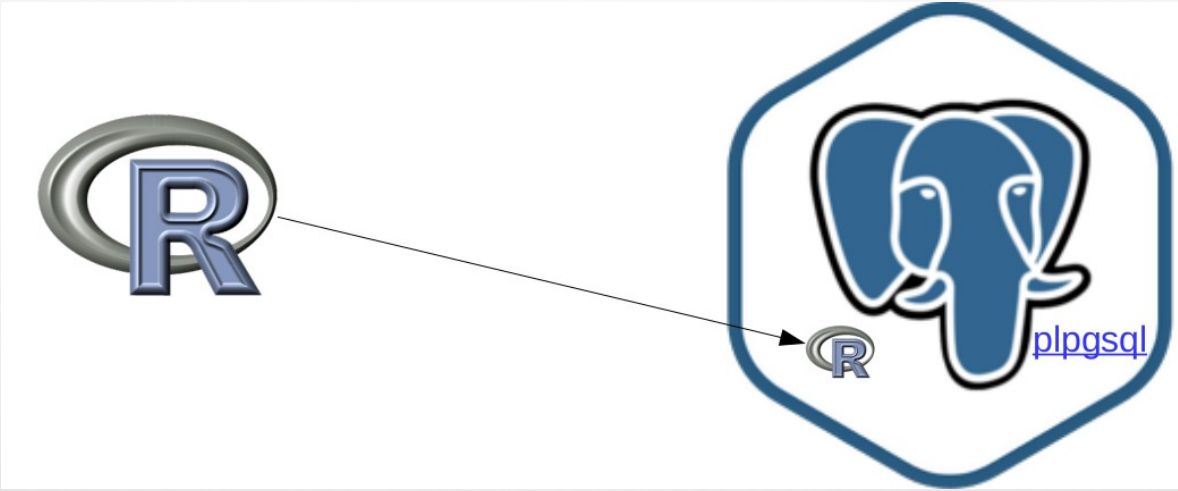
Rank	Change	Language	Share	Trend
1		Python	29.49 %	+4.5 %
2		Java	19.57 %	-2.4 %
3		Javascript	8.4 %	+0.1 %
4		C#	7.35 %	-0.4 %
5		PHP	6.34 %	-1.2 %
6		C/C++	5.87 %	-0.4 %
7		R	3.82 %	-0.2 %
8		Objective-C	2.6 %	-0.7 %
9		Swift	2.57 %	-0.1 %
10		Matlab	1.87 %	-0.2 %
11	↑	TypeScript	1.87 %	+0.3 %
12	↑↑↑↑	Kotlin	1.61 %	+0.6 %
13	↓↓	Ruby	1.47 %	-0.1 %
14	↓	VBA	1.39 %	-0.1 %
15	↑↑	Go	1.25 %	+0.3 %

https://www.tiobe.com/tiobe-index/
TIOBE INDEX

Oct 2019	Oct 2018	Change	Programming Language
1	1		Java
2	2		C
3	4	^	Python
4	3	v	C++
5	6	^	C#
6	5	v	Visual Basic .NET
7	8	^	JavaScript
8	9	^	SQL
9	7	v	PHP
10	15	^^	Objective-C
11	28	^^	Groovy
12	10	v	Swift
13	18	^^	Ruby
14	13	v	Assembly language
15	14	v	R

Características PLR en PostgreSQL

- PLR: es un lenguaje procedural “no confiable o desconfianza” para PostgreSQL que permite escribir funciones en el lenguaje R dentro del motor de bases de datos. Desarrollado por Joseph E. Conway desde el 2003



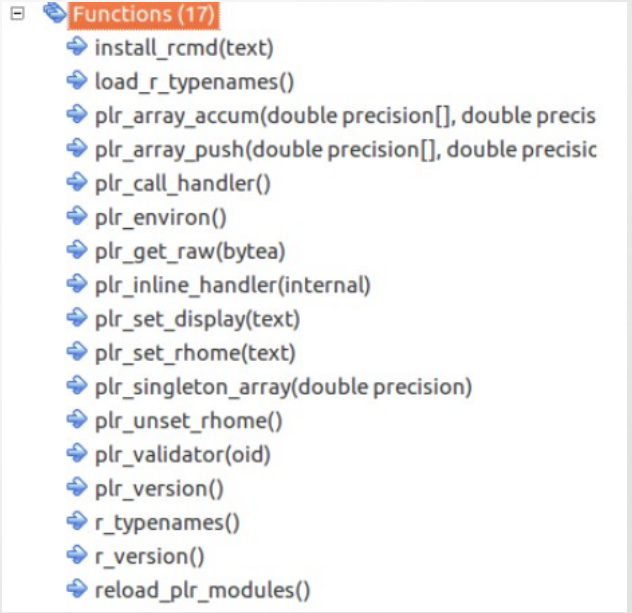
Características PLR en PostgreSQL

Beneficios de PLR:

- Hace disponible en PostgreSQL “cientos” o “miles” de funcionalidades estadísticas comprobadas y en desarrollo constante.
- Se aprovecha el amplio uso, conocimiento y documentación del lenguaje R para análisis de datos.
- Podría evitar saturación de la red cuando se analizan grandes volúmenes datos que se obtienen desde PostgreSQL

Características PLR en PostgreSQL

- Instalación de PLR:
  - Instalar R
    - sudo apt-get install r-base
  - Instalar PLR
    - sudo apt-get install postgresql-10-plr
    - CREATE EXTENSION plr;



Características PLR en PostgreSQL

Función en PLR: Se define igual que las demás funciones en PostgreSQL, con la sentencia **CREATE FUNCTION**. El cuerpo de la función es código R. El retorno de la función se realiza con la cláusula **RETURN**.

CREATE OR REPLACE FUNCTION nombre (argument-types)

RETURNS return-type AS $$

###Cuerpo de la función en R

$$

LANGUAGE plr ;

Características PLR en PostgreSQL

Función en PLR: Los parámetros de una función PL/R pueden nombrarse, debiendo ser llamados por dicho nombre dentro de la función, de no ser nombrados pueden ser accedidos con **argN**, siendo N el orden que ocupan en la lista de parámetros.

CREATE OR REPLACE FUNCTION suma(var1 integer, integer) RETURNS integer AS
$$
return (var1 + arg2)
$$ LANGUAGE plr;

```
log_analisis=# select suma(1,2);
 suma
------
    3
(1 fila)
```

```
log_analisis=# \df+ suma
                                                        Listado de funciones
 Esquema | Nombre | Tipo de dato de salida | Tipos de datos de argumentos | Tipo | Volatilidad | Paralelismo |  Dueño   | Seguridad | Privilegios | Lenguaje |    Código fuente     |
---------+--------+------------------------+------------------------------+------+-------------+-------------+----------+-----------+-------------+----------+----------------------+
 public  | suma   | integer                | var1 integer, integer        | func | volátil     | insegura    | postgres | invocador |             | plr      |                     +|
         |        |                        |                              |      |             |             |          |           |             |          | return (var1 + arg2)+|
```

Características PLR en PostgreSQL

- Homologación de tipos de datos

PostgreSQL	R
boolean	logical
int2 , int4	integer
int8, float4, float8, numeric	numeric
bytea	object
array	Vector (c)
tipo compuesto, funciones (table, setof)	dataframe
otro (char, text, character varying)	character

Ejemplos de funciones en PLR

- Una función en PLR

```
CREATE OR REPLACE   FUNCTION public.resumen_vector(vector int[])  RETURNS
TABLE(columna text) AS
$BODY$

   #condigo R
   df_vector <- as.data.frame(vector)
   return (summary(df_vector))

$BODY$
  LANGUAGE plr ;
```

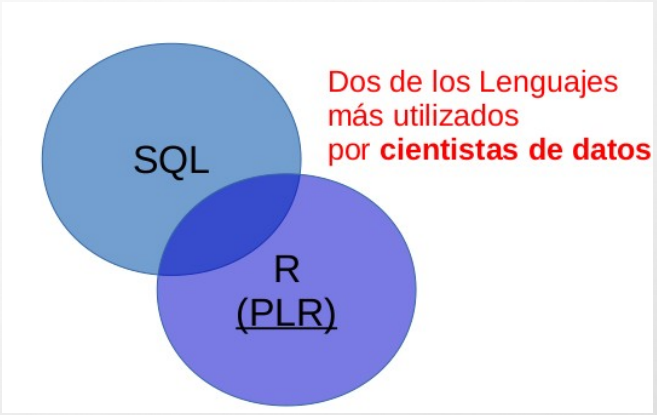
```
demos_r=#   select * from public.resumen_vector( array[4,2,3,4,5,6,3]);
     columna
-----------------
 Min.   :2.000
 1st Qu.:3.000
 Median :4.000
 Mean   :3.857
 3rd Qu.:4.500
 Max.   :6.000
(6 filas)

demos_r#
```

Ejemplos de funciones en PLR

- Una función en PLR

  select pais, **public.resumen_vector(vector)** from (**select country as pais, array_agg(age::integer) as vector from customers group by 1**) as datos

pais	resumen_vector
South Africa	Min. :18.00
South Africa	1st Qu.:36.00
South Africa	Median :54.00
South Africa	Mean :53.69
South Africa	3rd Qu.:72.00
South Africa	Max. :90.00
Australia	Min. :18.00
Australia	1st Qu.:34.00
Australia	Median :52.00
Australia	Mean :52.97
Australia	3rd Qu.:72.00
Australia	Max. :90.00
UK	Min. :18.0
UK	1st Qu.:37.0
UK	Median :55.0
UK	Mean :54.5
UK	3rd Qu.:72.0
UK	Max. :90.0

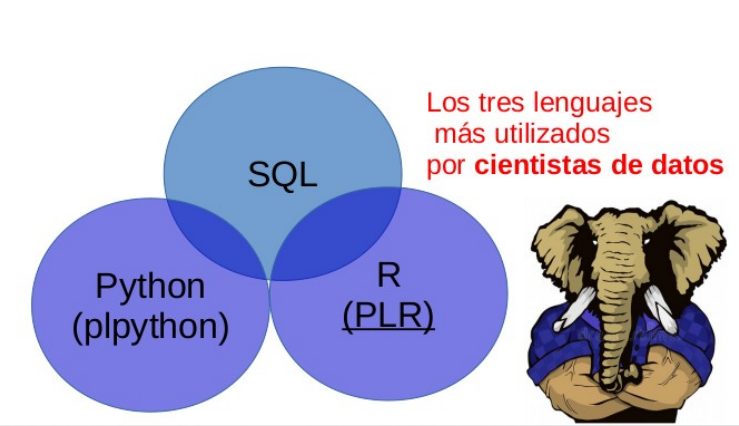


Ejemplos de funciones en PLR

- Una función en PLR

```
select pais, public.resumen_vector(vector ) from (select country as pais,
array_agg(age::integer) as vector from customers group by 1) as datos
```

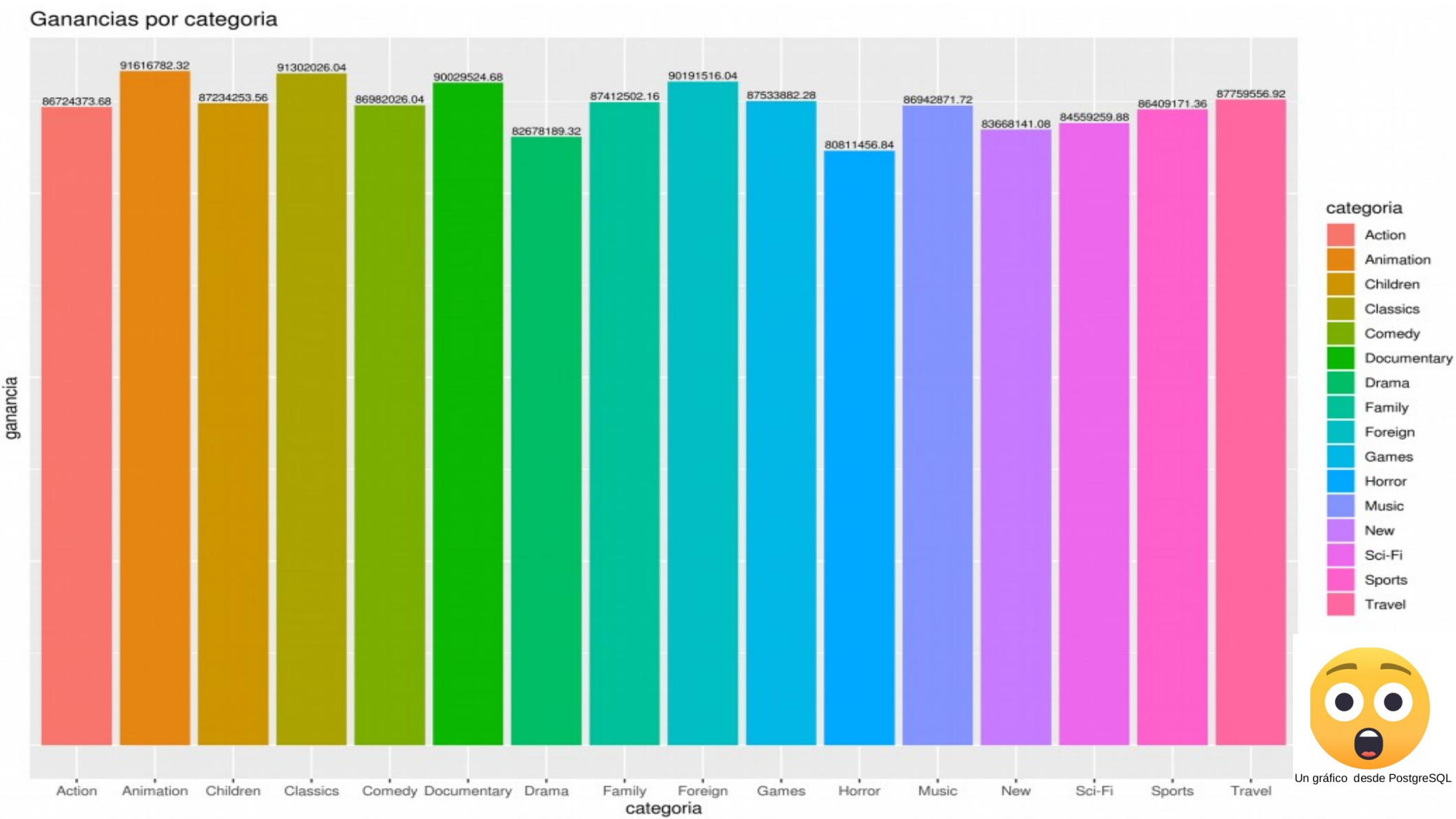
pais	resumen_vector
South Africa	Min. :18.00
South Africa	1st Qu.:36.00
South Africa	Median :54.00
South Africa	Mean :53.69
South Africa	3rd Qu.:72.00
South Africa	Max. :90.00
Australia	Min. :18.00
Australia	1st Qu.:34.00
Australia	Median :52.00
Australia	Mean :52.97
Australia	3rd Qu.:72.00
Australia	Max. :90.00
UK	Min. :18.0
UK	1st Qu.:37.0
UK	Median :55.0
UK	Mean :54.5
UK	3rd Qu.:72.0
UK	Max. :90.0



Ejemplos de funciones en PLR

- **pg.spi.exec(consulta):** donde consulta es una cadena de caracteres y la función devuelve los datos de un SELECT en un data.frame de R

```
CREATE OR REPLACE FUNCTION public.grafica_barras_cat2()   RETURNS text AS
$BODY$

require("ggplot2")
query <-"SELECT    categoria,   sum(ganancia) as ganancia FROM  resultado    group by 1;"

# hacer la query y enviar la salida a a resultado
resultado <- pg.spi.exec(query)

# hacer el grafico
g<-ggplot(data=resultado,aes(categoria,ganancia,fill=categoria))+ geom_bar(stat="identity")+
labs(title="Ganancias por categoria")+ geom_text(aes(label=ganancia),position = position_dodge(width = 1), vjust=-
0.3, size=2.5)+ theme(axis.text.y=element_blank(),  axis.ticks.y=element_blank())
ggsave("ganacias_cat.png", width = 12, height = 8)
return ("ganacias_cat.png");

$BODY$

  LANGUAGE plr;
```



Un gráfico desde PostgreSQL

Caso de uso- Análisis de logs

- Junto con las opciones del **Using CSV-Format Log Output**, se crearon varias funciones PLR que realizan **Text Mining** para descubrir información de los logs:
  - loga.gen_wordcloud
    - genera una nube de la palabras de los logs
  - loga.more_freq
    - para ver las palabras más frecuentes en los logs como una tabla
  - loga.gen_compare_words
    - Comparar dos palabras respecto a la ubicación en los logs y genera un grafico al respecto
  - loga.get_compare_words
    - Comparar dos palabras respecto a la ubicación en los logs como una tabla

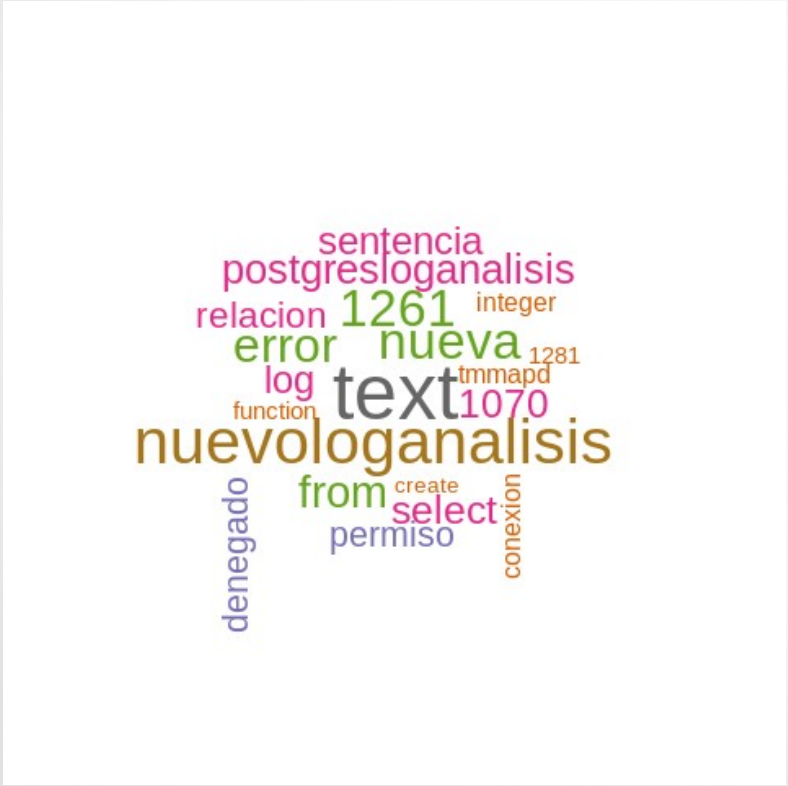
Caso de uso- Análisis de logs

loga.gen_wordcloud

```
 RETURNS void
 LANGUAGE plr
AS $function$
library(tm)
library(wordcloud)
txt <-  readLines(plog,encoding="UTF-8")
txt = iconv(txt, to="ASCII//TRANSLIT")
corpus <- Corpus(VectorSource(txt))
d  <- tm_map(corpus, content_transformer(tolower))
d  <- tm_map(d, stripWhitespace) #eliminar espacios en blancos
d <- tm_map(d, removePunctuation) #eliminar espacios en blancos
sw <- readLines("/home/anthony/vacias.txt",encoding="UTF-8")
sw = iconv(sw, to="ASCII//TRANSLIT")
d <- tm_map(d, removeWords, sw)
tdm <- TermDocumentMatrix(d)
###nube de palabras
m <- as.matrix(tdm)
v <- sort(rowSums(m),decreasing=TRUE)
df_fre <- data.frame(word = names(v),freq=v)

f_path <- pname
png(pname)
wordcloud(df_fre$word, df_fre$freq, min.freq = pfreq, random.order=FALSE,colors=brewer.pal(8, "Dark2"))
dev.off()

$function$
;
```

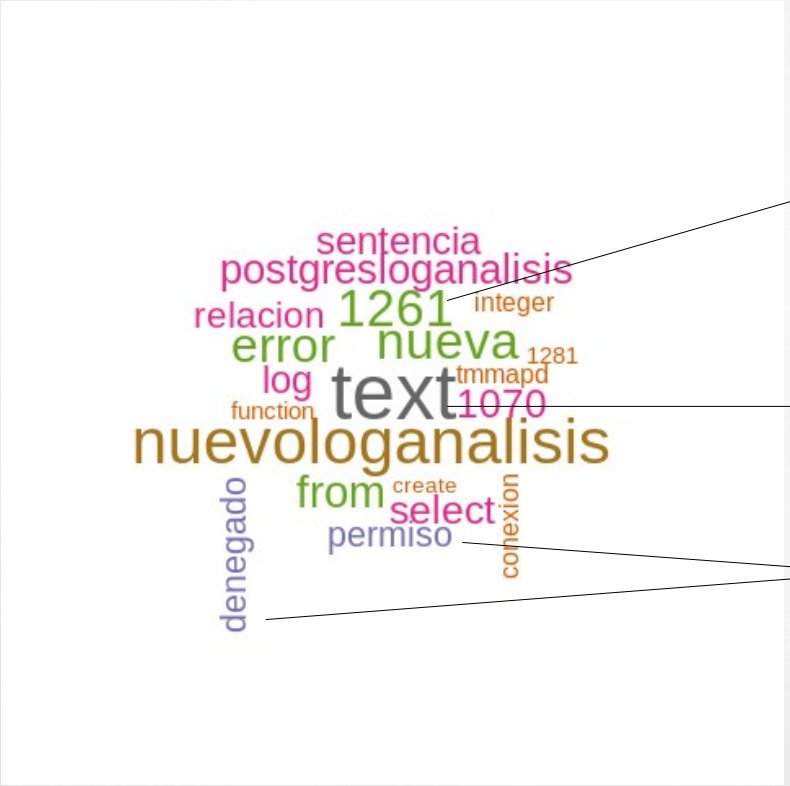
Caso de uso- Análisis de logs

```
select * from loga.gen_wordcloud ('/var/lib/postgresql/10/main/log/postgresql-2019-08-20_094238.log',20)
```



Caso de uso- Análisis de logs

```
select * from loga.gen_wordcloud ('/var/lib/postgresql/10/main/log/postgresql-2019-08-20_094238.log',20)
```



La sesión 1261 hizo muchas operaciones

Se crearon tablas con muchos atributos tipo text

Problemas de permisos denegados

Caso de uso- Análisis de logs

```
 RETURNS TABLE(word text, freq bigint)
 LANGUAGE plr
AS $function$
library(tm)
library(wordcloud)
txt <-  readLines(plog,encoding="UTF-8")
txt = iconv(txt, to="ASCII//TRANSLIT")
corpus <- Corpus(VectorSource(txt))
d  <- tm_map(corpus, content_transformer(tolower))
d  <- tm_map(d, stripWhitespace) #eliminar espacios en blancos
d <- tm_map(d, removePunctuation) #eliminar espacios en blancos
sw <- readLines("/home/anthony/vacias.txt",encoding="UTF-8")
sw = iconv(sw, to="ASCII//TRANSLIT")
d <- tm_map(d, removeWords, sw)
tdm <- TermDocumentMatrix(d)
m <- as.matrix(tdm)
v <- sort(rowSums(m),decreasing=TRUE)
df_fre <- data.frame(word = names(v),freq=v)
df_fre <- df_fre [ which(df_fre$freq>pfreq), ]

return (data.frame(df_fre$word,df_fre$freq))

$function$
;
```

loga.more_freq

Caso de uso- Análisis de logs

```
select * from  loga.more_freq('/var/lib/postgresql/10/main/log/postgresql-2019-08-20_094238.log',20);
```

	word	freq
1	text	114
2	nuevologanalisis	89
3	nueva	68
4	1261	68
5	error	64
6	from	57
7	postgresloganalisis	53
8	1070	50
9	select	50
10	log	48
11	sentencia	48
12	relacion	44
13	denegado	41
14	permiso	41
15	tmmapd	30
16	conexion	27
17	integer	27

Caso de uso- Análisis de logs

```
select * from  loga.more_freq('/var/lib/postgresql/10/main/log/postgresql-2019-08-20_094238.log',20);
```

	word	freq
1	text	114
2	nuevologanalisis	89
3	nueva	68
4	1261	68
5	error	64
6	from	57
7	postgresloganalisis	53
8	1070	50
9	select	50
10	log	48
11	sentencia	48
12	relacion	44
13	denegado	41
14	permiso	41
15	tmmapd	30
16	conexion	27
17	integer	27

Pero con FTS de PG
y to_tsvector se puede lograr esto y no tengo que ir a R

Caso de uso- Análisis de logs

```
select dat[1] word , array_length( string_to_array(dat[2],','),1) freq
from (
SELECT string_to_array
(unnest(string_to_array (to_tsvector('simple','postgresql postgresql postgresql
postgresql postgresql postgresql postgresql postgresql postgresql postgresql postgresql postgresql postgresql postgresql postgresql
postgresql postgresql postgresql postgresql postgresql postgresql postgresql postgresql postgresql postgresql postgresql postgresql
postgresql postgresql postgresql postgresql postgresql postgresql postgresql postgresql postgresql postgresql postgresql postgresql
postgresql postgresql postgresql postgresql postgresql postgresql postgresql postgresql postgresql postgresql postgresql postgresql
postgresql postgresql postgresql postgresql postgresql postgresql postgresql postgresql postgresql postgresql postgresql postgresql
postgresql postgresql postgresql postgresql postgresql postgresql postgresql postgresql postgresql postgresql postgresql postgresql
postgresql postgresql postgresql postgresql postgresql postgresql postgresql postgresql postgresql postgresql postgresql postgresql
postgresql postgresql postgresql postgresql postgresql postgresql postgresql postgresql postgresql postgresql postgresql postgresql
postgresql postgresql postgresql postgresql postgresql postgresql postgresql postgresql postgresql postgresql postgresql postgresql
postgresql postgresql postgresql postgresql postgresql postgresql postgresql postgresql postgresql postgresql postgresql postgresql
postgresql postgresql postgresql postgresql postgresql postgresql postgresql postgresql postgresql postgresql postgresql postgresql
postgresql postgresql postgresql postgresql postgresql postgresql postgresql postgresql postgresql postgresql postgresql postgresql
postgresql postgresql postgresql postgresql postgresql postgresql postgresql postgresql postgresql postgresql postgresql postgresql
postgresql postgresql postgresql postgresql postgresql postgresql postgresql postgresql postgresql postgresql postgresql postgresql
postgresql postgresql postgresql postgresql postgresql postgresql postgresql postgresql postgresql postgresql postgresql postgresql
postgresql postgresql postgresql postgresql postgresql postgresql postgresql postgresql postgresql postgresql postgresql postgresql
postgresql postgresql postgresql postgresql postgresql postgresql postgresql postgresql postgresql postgresql postgresql postgresql
postgresql postgresql postgresql postgresql postgresql postgresql postgresql postgresql postgresql postgresql postgresql postgresql
postgresql postgresql postgresql postgresql postgresql postgresql postgresql postgresql postgresql postgresql postgresql postgresql
postgresql postgresql postgresql postgresql postgresql postgresql postgresql postgresql postgresql postgresql postgresql postgresql
postgresql postgresql postgresql postgresql postgresql postgresql postgresql postgresql postgresql postgresql postgresql postgresql
postgresql postgresql postgresql postgresql postgresql postgresql postgresql postgresql postgresql postgresql postgresql postgresql
postgresql postgresql postgresql postgresql postgresql postgresql postgresql postgresql postgresql postgresql postgresql postgresql
postgresql postgresql postgresql postgresql postgresql postgresql postgresql postgresql postgresql postgresql postgresql postgresql
postgresql postgresql postgresql postgresql postgresql postgresql postgresql postgresql postgresql postgresql hola')::text, ' ')),':') as dat) as w order by 2
desc;
```

300 postgresql

word	freq
'postgresql'	255
'hola'	1

Máximo valor de freq 255

Caso de uso- Análisis de logs

Re: BUG #15600: ts_stat's nentry maxes out at 255

From: Tom Lane

Date: 21 January, 20:31:02

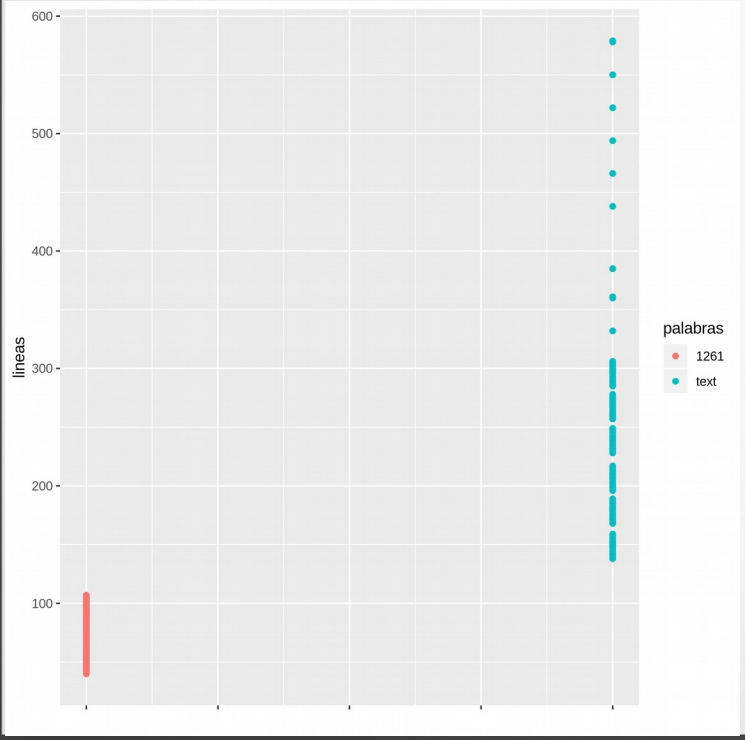
```
=?utf-8?q?PG_Bug_reporting_form?= <> writes:
> Unexpected behaviour:
> netry for 'hello' results in 255 despite 'hello' occurs 539 times in the
> attached test.

I think this is a consequence of the MAXNUMPOS limitation in the source
code, ie an individual tsvector won't store more than 255 locations for
the same word.  That's intentional to keep common words from bloating
tsvectors too much.  But if it's documented anywhere, I didn't see it.

            regards, tom lane
```

src/include/tsearch/ts_type.h
#define MAXNUMPOS (256)

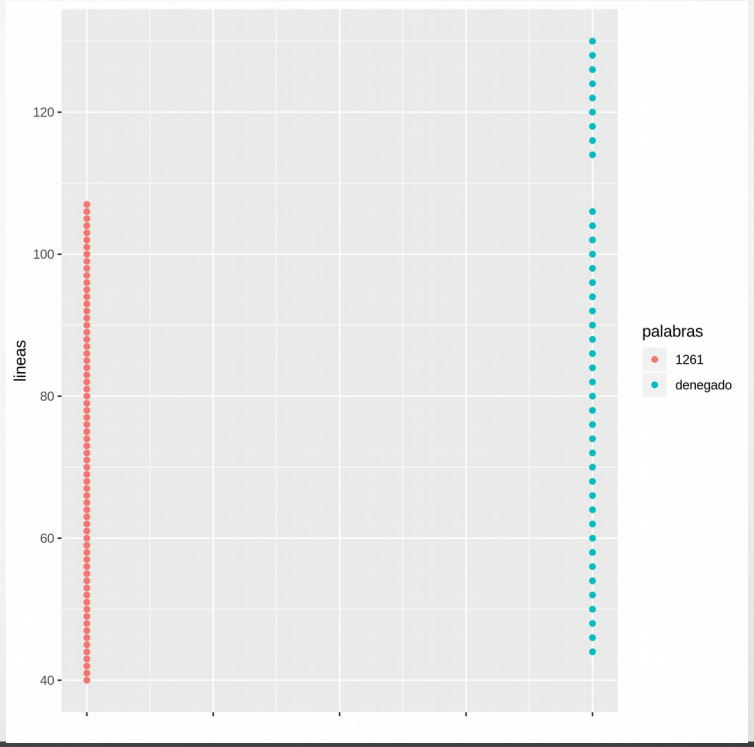
src/backend/utils/adt/tsvector.c
src/backend/utils/adt/tsvector_op.c
src/backend/tsearch/to_tsany.c

Caso de uso- Análisis de logs

```
select * from
loga.gen_compare_words
 ('/var/lib/postgresql/10/main/log/postgresql-2019-08-
20_094238.log',
'1261','text');
```




```
select * from
loga.gen_compare_words
('/var/lib/postgresql/10/main/log/postgresql-
2019-08-20_094238.log',
'1261','denegado');
```



Caso de uso- Análisis de logs

```
 select * from loga.get_compare_words
 ('/var/lib/postgresql/10/main/log/postgresql-
2019-08-20_094238.log',
'1261','denegado')
```

word1	word2
40	44
41	46
42	48
43	50
44	52
45	54
46	56
47	58
48	60
49	62
50	64
51	66
52	68
53	70
54	72
55	74
56	76
57	78
58	80
59	82
60	84
61	86
62	88
63	90

Caso de uso- Análisis de logs

```
select * from loga.get_compare_words
('/var/lib/postgresql/10/main/log/postgresql-2019-08-20_094238.log',
'1261','denegado')
```

word1	word2
40	44
41	46
42	48
43	50
44	52
45	54
46	56
47	58
48	60
49	62
50	64
51	66
52	68
53	70
54	72
55	74
56	76
57	78
58	80
59	82
60	84
61	86
62	88
63	90

```
with sub as  (
 select * from loga.get_compare_words
 ('/var/lib/postgresql/10/main/log/postgresql-2019-08-20_094238.log',
'1261','denegado')
)
select distinct sub1.word1, sub2.word2 from sub sub1 left join sub sub2 on (sub1.word1=sub2.word2)
```

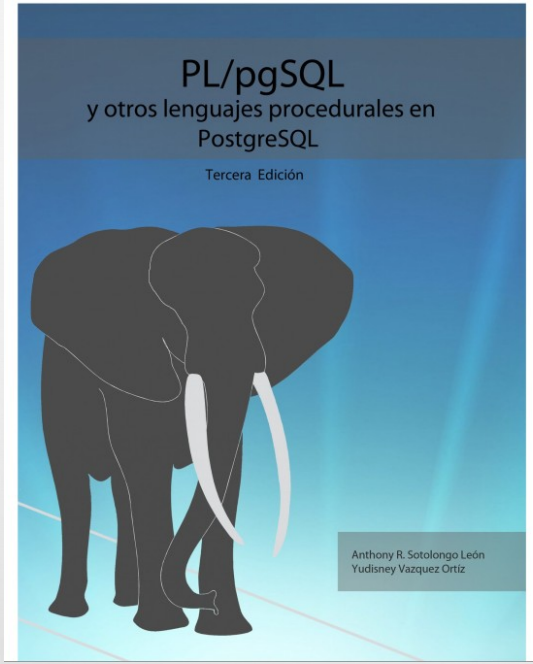
word1	word2
40	[NULL]
41	[NULL]
42	[NULL]
43	[NULL]
44	44
45	[NULL]
46	46
47	[NULL]
48	48
49	[NULL]
50	50
51	[NULL]
52	52
53	[NULL]
54	54
55	[NULL]

Caso de uso- Análisis de logs

- Recomendaciones para el uso de estas funciones:
  - Realizar el análisis en otro ***servidor distinto*** al de producción.
  - Mantener un ***log_rotation_size*** relativamente bajo para evitar archivos de logs muy grandes.
  - Definir las ***“palabras vacías”*** adecuadas.

Bibliografía

- http://www.joeconway.com/plr/doc/doc.html
- PL/pgSQL y otros lenguajes procedurales en PostgreSQL



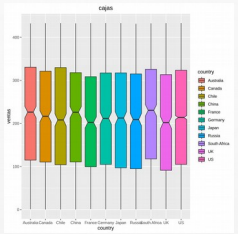
- **Capítulo 1.** Introducción a la programación del lado del servidor en PostgreSQL.
- **Capítulo 2.** Programación en SQL.
- **Capítulo 3.** Programación en PL/pgSQL.
- **Capítulo 4.** Análisis y perfilamiento de funciones y procedimientos.
- **Capítulo 5. Programación de funciones en lenguajes procedurales de desconfianza(plpython, PLR) de PostgreSQL**

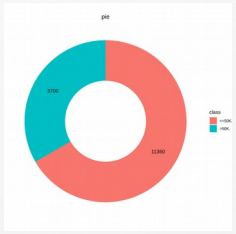
Trabajos “futuros”

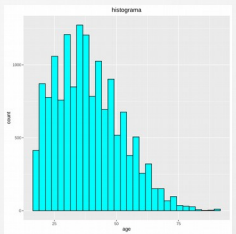
- Generación de gráficos desde PG
- Analítica de datos dentro de PG

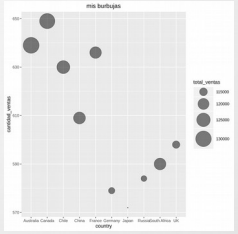
- Pie
- Histograma
- Puntos
- Cajas
- Burbujas
- Lineas
- Barras

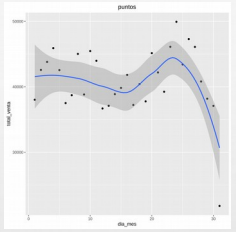
- Exploración de datos
- Análisis de Cluster
- Análisis de texto
- Análisis Predictivo
- Series de tiempo

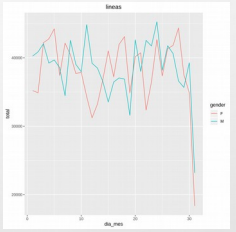


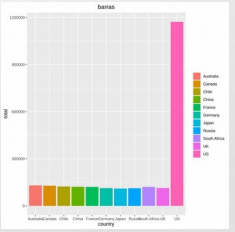








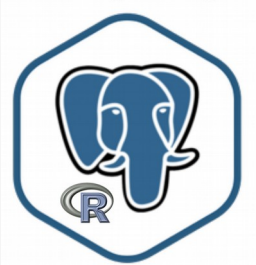




Conclusiones

- Se corrobora la **extensibilidad** de PostgreSQL, pues permite desarrollar funciones en otros lenguajes externos(en este caso R).
- PLR es la unión de dos proyectos de **opensource** R y PostgreSQL.
- El **Text Mining** puede resultar útil en el análisis de logs, pues por mucha experiencia que un DBA posea, esta técnica te **permite descubrir** información del comportamiento del servidor de BBDD que se nos puede pasar por alto.

PLR para análisis de logs en PostgreSQL(PG)



Anthony R. Sotolongo León
asotolongo@gmail.com

PgDay Santiago 2019