

6 Funciones SQL y plpgSQL

Links de interés:

PostgreSQL - BEGIN - GeeksforGeeks

A Computer Science portal for geeks. It contains well written, well thought and well explained computer science and

<https://geeksforgeeks.org/postgresql-begin/>

```
SET 1000 students (
  student_id,
  first_name,
  last_name,
  email,
  phone_number,
  hire_date,
  job_id,
  salary,
  commission_pct,
  manager_id,
  department_id
) = (
  SELECT * FROM students
  WHERE student_id < 1000
);
```

Capítulo 8 Modelo relacional | Base de Datos: Una aproximación a la información

Manual de referencia en bases de datos, desde un aspecto introductorio y forma p una serie de libros sobre datos e información.

<https://bookdown.org/paranadagarcia/database/modelo-relacional.html>

MOVE

MOVE MOVE — position a cursor
Synopsis MOVE [direction] [FROM | IN]
cursor_name where direction can ...

<https://www.postgresql.org/docs/current/sql-move.html>



PostgreSQL CHECK Constraint

In this tutorial, we will introduce you to PostgreSQL CHECK constraint to constrain the value of columns in a table based on a Boolean expression.

<https://postgresqtutorial.com/postgresql-tutorial/postgresql-check-constraint/>

CREATE TRIGGER

CREATE TRIGGER CREATE TRIGGER
— define a new trigger Synopsis CREATE
[OR REPLACE] [CONSTRAINT]

<https://www.postgresql.org/docs/current/sql-createtrigger.html>



Understanding PostgreSQL Triggers: A Comprehensive 101 Guide - Le

This article provides a comprehensive guide on PostgreSQL Triggers, different operations associated with them and the example queries to implement them.

<https://hevodata.com/learn/postgresql-triggers/>

Diferencias entre phpMyAdmin y Postgresql:

Regexp es ~

No usa comillas dobles

Los nombres de campo siempre con minúsculas

CONCAT es ||

Ampliación de conceptos:

Mirar lo que es una transacción en bases de datos (no lo ha explicado bien).

Mirar lo que es una variable tipo cursor en bases de datos

Vamos a dar hasta triggers

Mirar como blindar la base de datos con triggers

Conceptos básicos Postgresql

Podemos ver que se hacen consultas como en phpMyAdmin:

- Cuenta los empleados que su nombre empiece por G:

Query

Query History

1

SELECT COUNT(employeeenumber) FROM employees WHERE firstname LIKE 'G%'

Data Output

Messages

Notifications

count

bigint

1

3

- Dame todos los datos de los empleados que se llamen Jeff, Mary o Diane:

Query

Query History

Scratch Pad

1

SELECT * FROM employees e WHERE e.firstname ~ 'Jeff|Mary|Diane'

Data Output

Messages

Notifications

	employeenumber [FK] integer	lastname character varying (50)	firstname character varying (50)	extension character varying (10)	email character varying (100)	officecode character varying (10)	reportsto integer	jobtitle character varying (50)
1	1002	Murphy	Diane	x5800	dmurphy@classicmodelcars...	1	[null]	President
2	1056	Patterson	Mary	x4611	mpatterson@classicmodelcar...	1	1002	VP Sales
3	1076	Firrelli	Jeff	x9273	jfirrelli@classicmodelcars.com	1	1002	VP Marketing

- Dame todos los productos que empiecen por cuatro números en su nombre de producto:

Query

Query History

Scratch Pad

1

SELECT * FROM products p WHERE p.productName ~ '^[0-9]{4}.'

Data Output

Messages

Notifications

	productcode [PK] character varying (15)	productname character varying (70)	productline character varying (50)	productscale character varying (10)	productvendor character varying (50)	productdescription text
1	S10_1678	1969 Harley Davidson Ultimate Chopper	Motorcycles	1:10	Min Lin Diecast	This replica features working kickstand, front
2	S10_1949	1952 Alpine Renault 1300	Classic Cars	1:10	Classic Metal Creations	Turnable front wheels; steering function; deta
3	S10_2016	1996 Moto Guzzi 1100i	Motorcycles	1:10	Highway 66 Mini Classics	Official Moto Guzzi logos and insignias, sad

Empezamos Lenguaje Procedural

FUNCIONES:

Existen varios tipos de funciones pero solo vamos a dar dos tipos: funciones SQL y funciones en PL/pgSQL (plpgsql)

Funciones SQL: Son todas las funciones que solo haga operaciones SQL (CRUD: Create, Replace, Update o Delete; u otras operaciones SQL como Insert). Ej: *Función SQL que elimina los estudiantes de quinto año*

Funciones PL/pgSQL: Son todas las funciones que hagan operaciones empleando lenguajes de tipo procedural (Podemos resumirlo en todo lo que no sea CRUD). Ej: *Función en PL/pgSQL que suma 2 valores*

CREACIÓN DE FUNCIONES:

```
CREATE OR REPLACE FUNCTION nombreFunción( modoParámetro nombreParámetro/s tipoParámetro/s ) RETURNS
tipoDatoQueDevuelve/queTablaDevuelve AS
$$
Operación
$$ LANGUAGE tipoFunción ;
```

**El modoParámetro lo vamos a mirar más adelante, por ahora lo dejamos por defecto como IN*

Ej: Crear una función eliminar_estudiantes() que no devuelva nada (void) y que elimine todos los estudiantes de 5º año

```
CREATE OR REPLACE FUNCTION eliminar_estudiantes() RETURNS void AS
$$
DELETE FROM estudiante WHERE anno = 5;
$$ LANGUAGE sql;
```

El *nombreFunción()* es eliminar_estudiantes(); no existen parámetros para introducir; *tipoDatoQueDevuelve* es void por lo que no hace falta poner un RETURN; la Operación es un CRUD (Delete) por lo que el *tipoFunción* es LANGUAGE sql;

Ej: Crear una función sumar() que introduzcamos dos números enteros y nos devuelva su suma

```
CREATE OR REPLACE FUNCTION sumar(valor1 int, valor2 int) RETURNS int AS
$$
```

Empezamos creando una función *sumar()*, con *dos parámetros int* (CUIDADO CON EL ORDEN, ES A LA INVERSA DE JAVA) llamados valor1 y valor2 y nos va a *devolver un valor int* que como no es un valor void, tenemos que poner si o si el operador RETURN

```
BEGIN
RETURN $1 + $2;
END;
$$ LANGUAGE plpgsql;
```

Como sabemos que no es una operación CRUD, el **tipoFunción** es LANGUAGE plpgsql, por lo que después de los primeros \$\$ hay que indicar cuando empieza la operación con **BEGIN** y antes de los últimos \$\$ hay que indicar que finaliza la operación con **END**;

Para hacer la operación entre BEGIN y END, tenemos que indicar en algún punto lo que devuelve con **RETURN**. En este ejemplo como es una operación muy simple, solo tenemos que devolver la propia suma.

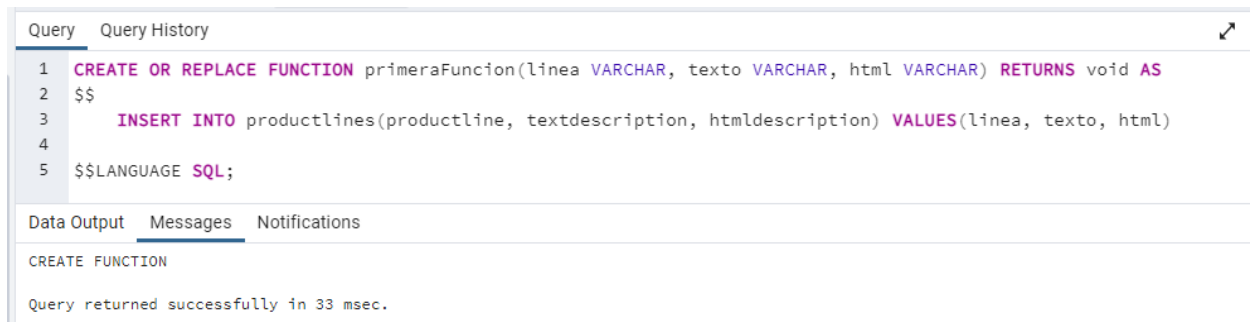
NOTA: Acuérdate que \$1 significa el primer parámetro (valor1) y \$2 el segundo parámetro (valor2) tienes que tener MUCHO cuidado porque el orden en plpgsql es bastante estricto.

NOTA: Intenta poner los parámetros con un nombre diferente a los atributos de las tablas o de las propias tablas

CREACIÓN Y EJECUCIÓN DE FUNCIONES SQL:

CREACIÓN:

Crea una función llamada primeraFuncion() que tengamos que escribir tres parámetros de texto (VARCHAR) que sean equivalentes a tres atributos de la tabla productlines (productline, textdescription, htmldescription), para insertar nuevos registros en productlines y que no nos devuelva nada

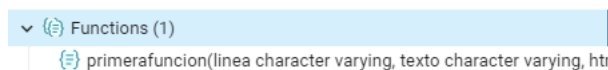


```
1 CREATE OR REPLACE FUNCTION primeraFuncion(linea VARCHAR, texto VARCHAR, html VARCHAR) RETURNS void AS
2 $$
3     INSERT INTO productlines(productline, textdescription, htmldescription) VALUES(linea, texto, html)
4
5 $$LANGUAGE SQL;
```

The screenshot shows a PostgreSQL query editor with a 'Query' tab selected. The query is a PL/pgSQL function definition. Below the query, the 'Messages' tab is selected, showing the message 'CREATE FUNCTION' and 'Query returned successfully in 33 msec.'

Como es un insert sabemos que es una operación SQL (**tipoFunción** es LANGUAGE SQL), por lo que no hay ni BEGIN ni END; el **nombreFunción()** es primeraFuncion(), los **nombreParámetros** son linea, texto y html; los **tipoParámetros** son todos VARCHAR; no devuelve nada así que el **tipoDatoQueDevuelve** es void. Por último hacemos un INSERT INTO como operación.

Ahora en Postgresql podemos buscar en el apartado de "Functions" y ver que está nuestra función creada:



NOTA: Si cambiamos el nombre, el tipo de RETURNS o los parámetros, se considera como otra [porque cambia la firma de la función], pero si cambiamos lo de dentro [la operación], se reemplazaría

NOTA: las funciones con parámetros diferentes pueden compartir el mismo nombre pero no las que tengan un RETURNS diferente pero el mismo nombre

EJECUCIÓN:

Usamos la función para ver si funciona, creando un nuevo registro con productline=Mini Motos, textdescription=Motos super pequeñas y htmldescription=null (devuelve VOID pero vemos que si que lo ha hecho)

Query Query History	
1	<code>SELECT primerafuncion('Mini motos', 'Motos super pequeñas', null)</code>
Data Output Messages Notifications	
<div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> </div>	
	<div> <div>primerafuncion</div> <div>void</div> <div></div> </div>
1	[null]

Ahora hacemos una consulta para ver si se ha creado un nuevo registro con Mini Motos; y efectivamente, es el registro número 8

Query

Query History

1

SELECT * FROM productlines

Data Output

Messages

Notifications

	productline [PK] character varying (50)	textdescription character varying (4000)
1	Classic Cars	Attention car enthusiasts: Make your v
2	Motorcycles	Our motorcycles are state of the art re
3	Planes	Unique, diecast airplane and helicopte
4	Ships	The perfect holiday or anniversary gift
5	Trains	Model trains are a rewarding hobby for
6	Trucks and Buses	The Truck and Bus models are realisti
7	Vintage Cars	Our Vintage Car models realistically pr
8	Mini motos	Motos super pequeñas

CREACIÓN:

Crea una función llamada primeraFuncion() que tengamos que escribir tres parámetros de texto (VARCHAR) que sean equivalentes a tres atributos de la tabla productlines (productline, textdescription, htmldescription), para insertar nuevos registros en productlines y que nos devuelva un texto concatenando el atributo productline con el textdescription dejando un espacio entre ambos

NOTA: Aunque estamos usando el mismo nombre de función y los mismos parámetros, al no usar el mismo tipo de RETURNS (antes era void y ahora es varchar) se entiende que es otra función diferente. Por lo que primero tenemos que borrar la función anterior y crear la nueva

Query Query History	
1	<code>CREATE OR REPLACE FUNCTION primeraFuncion(linea VARCHAR, texto VARCHAR, html VARCHAR) RETURNS VARCHAR AS</code>
2	<code>\$\$</code>
3	<code>INSERT INTO productlines(productline, textdescription, htmldescription) VALUES(linea, texto, html)</code>
4	<code>RETURNING productline ' ' textdescription;</code>
5	
6	<code>\$\$LANGUAGE SQL;</code>
Data Output Messages Notifications	
CREATE FUNCTION	
Query returned successfully in 43 msec.	

Todo es igual excepto el *tipoDatoQueDevuelve* ya que ahora en vez de devolver un void, devuelve VARCHAR (la cadena que concatena productline y textdescription) por lo que ahora sí que tenemos que hacer un **RETURNING**. El returning va a ser "productline || ' ' || textdescription" que básicamente es la concatenación (|| es el operador de CONCAT()) de productline+' '+textdescription

EJECUCIÓN:

Si hacemos un SELECT de la primerafuncion() con los parámetros que usamos en el anterior ejercicio, vemos que ahora en vez de devolver void, nos devuelve la cadena que compusimos y que pedimos que retornase con la sentencia de retorno

Query		Query History
1		SELECT primerafuncion('Maxi motos', 'Motos super grandes', null)
Data Output		Messages
		Notifications
	primerafuncion	
	character varying	
1	Maxi motos Motos super grandes	

CREACIÓN:

Crea una función llamada primeraFuncion() que tengamos que escribir tres parámetros de texto (VARCHAR) que sean equivalentes a tres atributos de la tabla productlines (productline, textdescription, htmldescription), para insertar nuevos registros en productlines y que nos devuelva un texto concatenando el atributo productline con el textdescription dejando un espacio entre ambos con una coma

NOTA: Como ahora escribimos lo mismo (nombre de función, parámetros y el RETURNS son iguales) pero cambiando algo entre los dólares, vemos que no se crea una nueva función, sino que se reemplaza la antigua primeraFuncion por la que creamos ahora

Query		Query History
1		CREATE OR REPLACE FUNCTION primeraFuncion(linea VARCHAR, texto VARCHAR, html VARCHAR) RETURNS VARCHAR AS
2		\$\$
3		INSERT INTO productlines(productline, textdescription, htmldescription) VALUES (linea, texto, html)
4		RETURNING productline ' ' , ' ' textdescription;
5		
6		\$\$LANGUAGE SQL;
Data Output		Messages
		Notifications
		CREATE FUNCTION
		Query returned successfully in 37 msec.

Ahora en el RETURNING en vez de concatenar ' ' se concatena ' , ' entre productline y textdescription

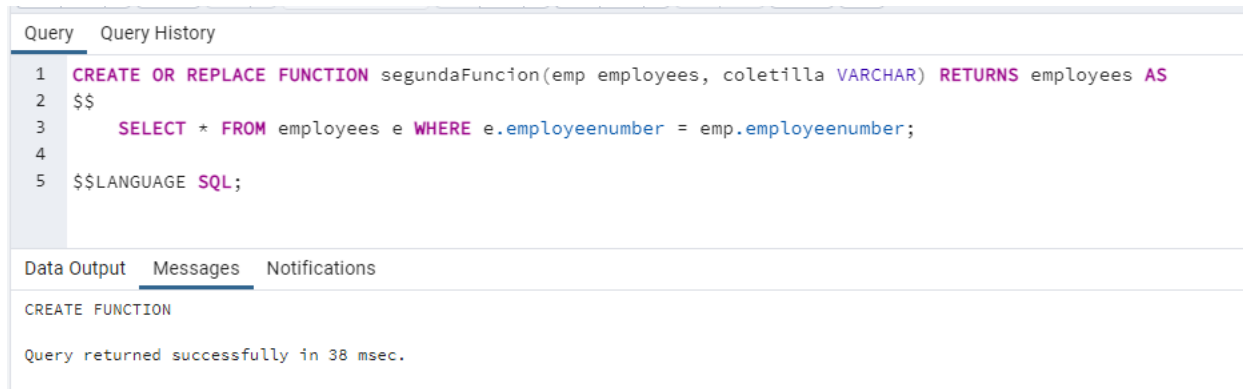
Podemos ver la función reemplazada ahora:

▼	Functions (1)
	primerafuncion(linea character varying, texto character varying, htr

CREACIÓN:

Creamos una función llamada segundaFuncion a la que tengamos que pasarle como parámetros la tabla employees y una cadena de texto y nos devuelva la misma tabla como un objeto.

En este caso recibe un registro de employees y lo devuelve:



```
Query  Query History
1  CREATE OR REPLACE FUNCTION segundaFuncion(emp employees, coletilla VARCHAR) RETURNS employees AS
2  $$
3      SELECT * FROM employees e WHERE e.employeenumber = emp.employeenumber;
4
5  $$LANGUAGE SQL;
```

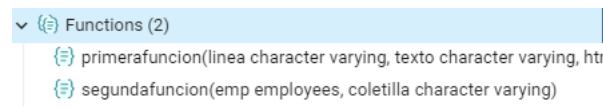
Data Output Messages Notifications

CREATE FUNCTION

Query returned successfully in 38 msec.

Ahora se crea la función segundaFuncion con una tabla emp que hace referencia a employees y una cadena de texto llamada coletilla. Nos va a devolver la tabla employees (Hay que tener en cuenta de que cuando hacer RETURN + tabla, va a tratar a una tabla como un objeto y *no hace falta poner un RETURNING en la operación ya que al usar una operación SELECT no es necesario*). La operación es una operación SQL ya que hacemos que el atributo employeenumber de la tabla employees se iguale al mismo atributo de la tabla emp que es nuestro parámetro.

Ahora vemos que se ha creado la función:



EJECUCIÓN:

(ponemos el hola porque pusimos que teníamos que pasarle un registro y una string)

Ahora al escribir la sentencia SELECT usando la función y el FROM haciendo referencia a la tabla employees, nos devuelve todos los parámetros de employees y en cada fila sus atributos separados con comas, como un objeto.

NOTA: Si usásemos cualquier otra tabla no funcionaría porque solo la tabla employees tiene el atributo employeenumber

*NOTA: Para referirte a una tabla como parámetro de una función si estamos ejecutando esa función, tenemos que poner la tabla (o el alias) a la que hace referencia el FROM y seleccionar todos sus atributos con la sentencia: tabla.**

Query Query History	
1	SELECT segundafuncion(e.*, 'hoLa') FROM employees e
Data Output Messages Notifications	
<div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> </div>	
	segundafuncion employees
1	(1002,Murphy,Diane,x5800,dmurphy@classicmodelcars.com,1,,President)
2	(1056,Patterson,Mary,x4611,mpatterso@classicmodelcars.com,1,1002,"VP Sales")
3	(1076,Firrelli,Jeff,x9273,jfirrelli@classicmodelcars.com,1,1002,"VP Marketing")
4	(1088,Patterson,William,x4871,wpatterson@classicmodelcars.com,6,1056,"Sales Manager (APA...
5	(1102,Bondur,Gerard,x5408,gbondur@classicmodelcars.com,4,1056,"Sale Manager (EMEA)")
6	(1143,Bow,Anthony,x5428,abow@classicmodelcars.com,1,1056,"Sales Manager (NA)")
7	(1165,Jennings,Leslie,x3291,ljennings@classicmodelcars.com,1,1143,"Sales Rep")
8	(1166,Thompson,Leslie,x4065,lthompson@classicmodelcars.com,1,1143,"Sales Rep")
9	(1188,Firrelli,Julie,x2173,jfirrelli@classicmodelcars.com,2,1143,"Sales Rep")
10	(1216,Patterson,Steve,x4334,spatterson@classicmodelcars.com,2,1143,"Sales Rep")
11	(1286,Tseng,"Foon Yue",x2248,ftseng@classicmodelcars.com,3,1143,"Sales Rep")
12	(1323,Vanauf,George,x4102,gvanauf@classicmodelcars.com,3,1143,"Sales Rep")
13	(1337,Bondur,Loui,x6493,lbondur@classicmodelcars.com,4,1102,"Sales Rep")

CREACIÓN:

Reemplaza la segundaFunción para que ahora haga un update en el atributo jobtitle de la tabla employees haciendo que se concatene el jobtitle y el parámetro de texto con un espacio entre ambos

NOTA: Cuando hacemos sentencias de UPDATE, INSERT, DELETE si que hay que poner un RETURNING, mientras que si usamos un SELECT, no hace falta el RETURNING

Query Query History	
1	CREATE OR REPLACE FUNCTION segundaFuncion(emp employees, coletilla VARCHAR) RETURNS employees AS
2	\$\$
3	UPDATE employees SET jobtitle=jobtitle ' ' coletilla WHERE employeenumber=emp.employeenumber
4	RETURNING employees.*;
5	
6	\$\$ LANGUAGE SQL ;
Data Output Messages Notifications	
CREATE FUNCTION	
Query returned successfully in 72 msec.	

Al ser un reemplazo, sabemos que tenemos que mantener el nombre de la función, sus parámetros y el RETURNS, por lo que tenemos que cambiar la operación únicamente. Al ser un Update es una operación SQL ya que es una operación CRUD. En el UPDATE vemos que se hace un UPDATE de la tabla employees en donde ahora el jobtitle es igual a jobtitle+' '+coletilla (gracias al operador de concatenación ||) cuando el employeenumber de la tabla employee es igual al employeenumber de nuestra tabla parámetro emp. Por último se hace el RETURNING de todos los atributos de la tabla employees.

NOTA: Cuidado con los ; porque si lo ponemos al final de la sentencia UPDATE, no nos hace el método

EJECUCIÓN

Ahora modificamos todos los Sales Rep para que sean SENIOR y nos devuelve los empleados cambiados

Query		Query History
1 SELECT segundafuncion(e.*, 'SENIOR') FROM employees e		
2 WHERE jobtitle='Sales Rep'		
Data Output		Messages Notifications
	segundafuncion employees	
1	(1165,Jennings,Leslie,x3291,ljennings@classicmodelcars.com,1,1143,"Sales Rep SENIOR")	
2	(1166,Thompson,Leslie,x4065,lthompson@classicmodelcars.com,1,1143,"Sales Rep SENIO...	
3	(1188,Firrelli,Julie,x2173,jfirrelli@classicmodelcars.com,2,1143,"Sales Rep SENIOR")	
4	(1216,Patterson,Steve,x4334,spatterson@classicmodelcars.com,2,1143,"Sales Rep SENIOR")	
5	(1286,Tseng,"Foon Yue",x2248,ftseng@classicmodelcars.com,3,1143,"Sales Rep SENIOR")	
6	(1323,Vanauf,George,x4102,gvanauf@classicmodelcars.com,3,1143,"Sales Rep SENIOR")	
7	(1337,Bondur,Loui,x6493,lbondur@classicmodelcars.com,4,1102,"Sales Rep SENIOR")	

CREACIÓN:

Crea una función llamada buscarempleado que al pasar un mail, nos devuelva todos los datos de los empleados que contengan ese nombre

Query		Query History
1 CREATE OR REPLACE FUNCTION buscarempleado(mail varchar) RETURNS employees AS		
2 \$\$		
3 SELECT e.* FROM employees e WHERE e.email=mail;		
4 \$\$ LANGUAGE SQL ;		
5		
Data Output		Messages Notifications
CREATE FUNCTION		
Query returned successfully in 116 msec.		

Creamos una función SQL buscarempleado al que le pasemos un parámetro y nos devuelva los employees.

```
1 CREATE OR REPLACE FUNCTION buscarempleados(emp employees, mail varchar) RETURNS employees AS
2 $$
3 SELECT e.* FROM employees e WHERE e.email=mail AND emp.employeenumber=employeenumber
4 $$LANGUAGE SQL;
```

~~~~~

## EJECUCIÓN:

Ahora vamos a buscar un mail:

|   |                     |                                                              |
|---|---------------------|--------------------------------------------------------------|
| 6 | <code>SELECT</code> | <code>buscarepleado('jfirrelli@classicmodelcars.com')</code> |
|---|---------------------|--------------------------------------------------------------|

| Data Output                                                                                                            | Messages                                                                        | Notifications |
|------------------------------------------------------------------------------------------------------------------------|---------------------------------------------------------------------------------|---------------|
| <div> <div>+</div> <div>📄</div> <div>▼</div> <div>📋</div> <div>🗑️</div> <div>🔍</div> <div>⬇️</div> <div>📈</div> </div> |                                                                                 |               |
| <div> <div>buscarepleado</div> <div>employees</div> <div>🔒</div> </div>                                                |                                                                                 |               |
| 1                                                                                                                      | (1076,Firrelli,Jeff,x9273,jfirrelli@classicmodelcars.com,1,1002,"VP Marketin... |               |

Vemos que nos aparece la primera persona que tiene un mail como jfirrelli@classicmodelcars.com, que es Jeff, pero no nos aparece la otra persona con el mismo mail, porque para eso tenemos que hacer un loop, que lo veremos más adelante. Lo importante de esta consulta es entender que únicamente nos va a devolver un resultado

*NOTA: Cuidado con los " porque si no los ponemos entre el parámetro que estamos utilizando nos da este error:*

|   |                     |                                                            |
|---|---------------------|------------------------------------------------------------|
| 6 | <code>SELECT</code> | <code>buscarepleado(jfirrelli@classicmodelcars.com)</code> |
| 7 |                     |                                                            |

| Data Output                                                                                                                                                     | Messages | Notifications |
|-----------------------------------------------------------------------------------------------------------------------------------------------------------------|----------|---------------|
| <div> <div>+</div> <div>📄</div> <div>▼</div> <div>📋</div> <div>🗑️</div> <div>🔍</div> <div>⬇️</div> <div>📈</div> </div>                                          |          |               |
| <div> <div>ERROR: no existe la columna «jfirrelli»</div> <div>LINE 1: SELECT buscarpleado(jfirrelli@classicmodelcars.com)</div> <div></div> <div>A</div> </div> |          |               |
| <div> <div>SQL state: 42703</div> <div>Character: 23</div> </div>                                                                                               |          |               |

*NOTA: Si pones FROM employees, vas a hacer que te devuelva el resultado de la primera persona que tiene el correo jfirrelli@classicmodelcars.com replicado 23 veces que son los 23 registros que hay en la tabla employees, así que cuidado*

|   |                     |                                                              |                   |                        |
|---|---------------------|--------------------------------------------------------------|-------------------|------------------------|
| 6 | <code>SELECT</code> | <code>buscarepleado('jfirrelli@classicmodelcars.com')</code> | <code>FROM</code> | <code>employees</code> |
|---|---------------------|--------------------------------------------------------------|-------------------|------------------------|

| Data Output                                                                                                            | Messages                                                                        | Notifications |
|------------------------------------------------------------------------------------------------------------------------|---------------------------------------------------------------------------------|---------------|
| <div> <div>+</div> <div>📄</div> <div>▼</div> <div>📋</div> <div>🗑️</div> <div>🔍</div> <div>⬇️</div> <div>📈</div> </div> |                                                                                 |               |
| <div> <div>buscarepleado</div> <div>employees</div> <div>🔒</div> </div>                                                |                                                                                 |               |
| 13                                                                                                                     | (1076,Firrelli,Jeff,x9273,jfirrelli@classicmodelcars.com,1,1002,"VP Marketin... |               |
| 14                                                                                                                     | (1076,Firrelli,Jeff,x9273,jfirrelli@classicmodelcars.com,1,1002,"VP Marketin... |               |
| 15                                                                                                                     | (1076,Firrelli,Jeff,x9273,jfirrelli@classicmodelcars.com,1,1002,"VP Marketin... |               |
| 16                                                                                                                     | (1076,Firrelli,Jeff,x9273,jfirrelli@classicmodelcars.com,1,1002,"VP Marketin... |               |
| 17                                                                                                                     | (1076,Firrelli,Jeff,x9273,jfirrelli@classicmodelcars.com,1,1002,"VP Marketin... |               |
| 18                                                                                                                     | (1076,Firrelli,Jeff,x9273,jfirrelli@classicmodelcars.com,1,1002,"VP Marketin... |               |
| 19                                                                                                                     | (1076,Firrelli,Jeff,x9273,jfirrelli@classicmodelcars.com,1,1002,"VP Marketin... |               |
| 20                                                                                                                     | (1076,Firrelli,Jeff,x9273,jfirrelli@classicmodelcars.com,1,1002,"VP Marketin... |               |
| 21                                                                                                                     | (1076,Firrelli,Jeff,x9273,jfirrelli@classicmodelcars.com,1,1002,"VP Marketin... |               |
| 22                                                                                                                     | (1076,Firrelli,Jeff,x9273,jfirrelli@classicmodelcars.com,1,1002,"VP Marketin... |               |
| 23                                                                                                                     | (1076,Firrelli,Jeff,x9273,jfirrelli@classicmodelcars.com,1,1002,"VP Marketin... |               |

## CREACIÓN:

Primero creamos un nuevo registro:

```

1 INSERT INTO customers (customernumber, customername, contactlastname, contactfir
2     VALUES (200, 'Programación', 'Prado', 'Fernando', '6666666666', '25, rúa Coru
3
4 select * from customers

```

Data Output

Messages

Notifications

|     | customernumber<br>[PK] integer | customername<br>character varying (50) | contactlastname<br>character varying (50) | contactfirstname<br>character varying (50) | phone<br>character varying (50) | addressline1<br>character varying ( |
|-----|--------------------------------|----------------------------------------|-------------------------------------------|--------------------------------------------|---------------------------------|-------------------------------------|
| 115 | 480                            | Kremlin Collectables, Co.              | Semenov                                   | Alexander                                  | +7 812 293 0521                 | 2 Pobeuy Square                     |
| 116 | 481                            | Raanan Stores, Inc                     | Altagar,G M                               | Raanan                                     | + 972 9 959 8555                | 3 Hagalim Blv.                      |
| 117 | 484                            | Iberia Gift Imports, Corp.             | Roel                                      | José Pedro                                 | (95) 555 82 82                  | C/ Romero, 33                       |
| 118 | 486                            | Motor Mint Distributors Inc.           | Salazar                                   | Rosa                                       | 2155559857                      | 11328 Douglas Av                    |
| 119 | 487                            | Signal Collectibles Ltd.               | Taylor                                    | Sue                                        | 4155554312                      | 2793 Furth Circle                   |
| 120 | 489                            | Double Decker Gift Stores, Ltd         | Smith                                     | Thomas                                     | (171) 555-7555                  | 120 Hanover Sq.                     |
| 121 | 495                            | Diecast Collectables                   | Franco                                    | Valarie                                    | 6175552555                      | 6251 Ingle Ln.                      |
| 122 | 496                            | Kellys Gift Shop                       | Snowden                                   | Tony                                       | +64 9 5555500                   | Arenales 1938 3A                    |
| 123 | 200                            | Programación                           | Prado                                     | Fernando                                   | 6666666666                      | 25, rúa Coruña                      |

Ahora hacemos una función para eliminar a todos los clientes de Coruña:

| Query | Query History                                                                    |
|-------|----------------------------------------------------------------------------------|
| 1     | <b>CREATE OR REPLACE FUNCTION</b> eliminarCoruña() <b>RETURNS</b> void <b>AS</b> |
| 2     | \$\$                                                                             |
| 3     | <b>DELETE FROM</b> customers <b>WHERE</b> city='A Coruña';                       |
| 4     | \$\$ <b>LANGUAGE sql</b> ;                                                       |









| Data Output                                                                                      | Messages | Notifications |
|--------------------------------------------------------------------------------------------------|----------|---------------|
| <div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> <div></div> </div> |          |               |
| CREATE FUNCTION                                                                                  |          |               |
| Query returned successfully in 76 msec.                                                          |          |               |


### EJECUCIÓN:

Ejecutamos la consulta:

```
6 SELECT eliminarCoruña()
```

Data Output   Messages   Notifications

|   |                                                                                                                   |
|---|-------------------------------------------------------------------------------------------------------------------|
|   | <b>eliminarcoruña</b><br>void  |
| 1 | [null]                                                                                                            |

Y ahora comprobamos que está todo correcto y hemos borrado el registro de Coruña:

8 **SELECT \* FROM** customers

Data Output Messages Notifications

|     | customernumber<br>[PK] integer | customername<br>character varying (50) | contactlastname<br>character varying (50) | contactfirstname<br>character varying (50) | phone<br>character |
|-----|--------------------------------|----------------------------------------|-------------------------------------------|--------------------------------------------|--------------------|
| 112 | 473                            | Frau da Collezione                     | Ricotti                                   | Franco                                     | +39 0225           |
| 113 | 475                            | West Coast Collectables Co.            | Thompson                                  | Steve                                      | 31055537           |
| 114 | 477                            | Mit Vergnügen & Co.                    | Moos                                      | Hanna                                      | 0621-085           |
| 115 | 480                            | Kremlin Collectables, Co.              | Semenov                                   | Alexander                                  | +7 812 29          |
| 116 | 481                            | Raanan Stores, Inc                     | Altagar,G M                               | Raanan                                     | + 972 9 9          |
| 117 | 484                            | Iberia Gift Imports, Corp.             | Roel                                      | José Pedro                                 | (95) 555 1         |
| 118 | 486                            | Motor Mint Distributors Inc.           | Salazar                                   | Rosa                                       | 21555598           |
| 119 | 487                            | Signal Collectibles Ltd.               | Taylor                                    | Sue                                        | 41555543           |
| 120 | 489                            | Double Decker Gift Stores, Ltd         | Smith                                     | Thomas                                     | (171) 555          |
| 121 | 495                            | Diecast Collectables                   | Franco                                    | Valarie                                    | 61755525           |
| 122 | 496                            | Kellys Gift Shop                       | Snowden                                   | Tony                                       | +64 9 555          |

*NOTA: En este caso, si hubiéramos creado tres nuevos registros de Coruña, si que nos hubiera borrado los tres y no solo el primero. Hay bastante diferencia entre crear una función que afecte a varios registros (que entonces afecta a esos registros) y ejecutar la función para que te devuelva varios registros*

#### CREACIÓN:

Volvemos a crear registros de Coruña, esta vez 3

Ahora vamos a crear una función que además de eliminar los clientes de Coruña, se muestre la cantidad de clientes restantes. Como a la función existente no se le puede cambiar el tipo de RETURNS (pasamos de void a bigint), se debe eliminar e implementar una nueva con las nuevas operaciones

```

1 CREATE OR REPLACE FUNCTION eliminarCoruña() RETURNS bigint AS
2 $$
3 DELETE FROM customers WHERE city='A Coruña';
4 SELECT count(*) FROM customers;
5 $$ LANGUAGE sql;
6
Data Output Messages Notifications
CREATE FUNCTION
Query returned successfully in 53 msec.
```

La sentencia DELETE es igual que la del anterior ejemplo pero la sentencia SELECT es la que utilizamos para que nos devuelva la cantidad de clientes restantes, por lo que ya no tenemos que poner un RETURNING

*NOTA: Se pueden hacer todas las operaciones que queramos entre los \$\$*

#### EJECUCIÓN:

Ahora vemos que en vez de devolver void, nos devuelve el total de clientes sin los de Coruña.

|   |        |                  |    |                        |
|---|--------|------------------|----|------------------------|
| 1 | SELECT | eliminarcoruña() | as | totalCLientesSinCoruña |
| 2 |        |                  |    |                        |

---

|             |          |               |
|-------------|----------|---------------|
| Data Output | Messages | Notifications |
|-------------|----------|---------------|

---

|                        |        |
|------------------------|--------|
| totalclientessincoruña | bigint |
| 1                      | 122    |

## TEORÍA DE LAS FUNCIONES:

Habíamos hablado de que la sintaxis para crear una función es la siguiente:

```
CREATE OR REPLACE FUNCTION nombreFunción( modoParámetro nombreParámetro/s tipoParámetro/s ) RETURNS
tipoDatoQueDevuelve/queTablaDevuelve AS
$$
    Operación
$$LANGUAGE tipoFunción ;
```

El modoParámetro puede ser de 4 tipos:

- **IN:** Es el modo por defecto por lo que no hay que escribirlo. Especifica que el parámetro es de entrada, o sea, forma parte de la lista de parámetros con que se invoca a la función y que son necesarios para el procesamiento definido en la función.
- **OUT:** Es el parámetro de salida, forma parte del resultado de la función y no se incluye en la invocación de la función.
- **INOUT:** Es el parámetro de entrada/salida, puede ser empleado indistintamente para que forme parte de la lista de parámetros de entrada y que sea parte luego del resultado.
- **VARIADIC:** Es el parámetro de entrada con un tratamiento especial, que permite definir un arreglo para especificar que la función acepta un conjunto variable de parámetros, los que lógicamente deben ser del mismo tipo.

*\*Lo vamos a ver con más detalle en el siguiente PDF una vez demos operadores*

### Otras cosas que tener en cuenta

-Intentar siempre darles nombres tanto a los parámetros como a cualquier otro dato de la función, diferentes a los nombres de las tablas o de sus atributos

-Los parámetros pueden ser referenciados con números así: **\$número**, refiriéndose \$1 al primer parámetro definido, \$2 al segundo, y así sucesivamente. Esto funciona tanto si le hemos dado un nombre al parámetro como si no.

-Si un parámetro es de tipo compuesto se puede emplear la notación parámetro.campo para acceder a sus atributos.

-Las funciones SQL más simples no tienen parámetros o retornan un tipo de dato básico, por lo que el retorno que se puede realizar:

- Utilizando una consulta SELECT como la última del bloque de sentencias SQL de la función.
- Empleando la cláusula RETURNING como parte de las consultas INSERT, UPDATE o DELETE y todas las de plpgsql