

HITO 2 DEL 2º TRIMESTRE DE BASE DE DATOS

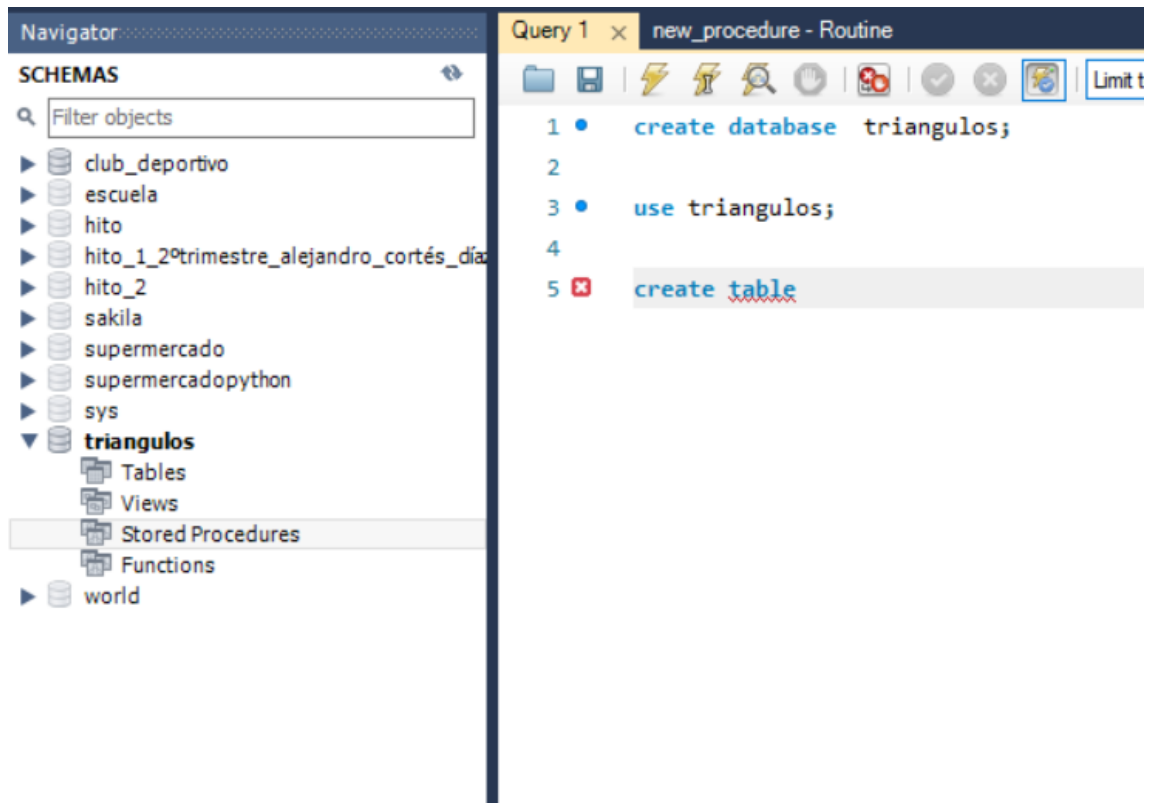
Alejandro Cortés Díaz

Índice

Programación con MySQL (5 puntos en total)	2
1. Crea una base de datos llamada triángulos.	3
2. Crea un procedimiento almacenado dentro de la base de datos triángulos que realice las siguientes acciones:	3
a. Crea una tabla llamada triángulo con los campos lado1, lado2 y lado3. Si la tabla ya existe, se borrará y se volverá a crear de nuevo.	4
b. Añade 20 filas a la tabla triángulo con valores al azar entre 1 y 5 para cada uno de los lados.	4
3. Crea una función PL/SQL que reciba tres números enteros (los tres lados del triángulo) y retorne una cadena indicando si el triángulo es Equilátero, Isósceles o Escaleno.	6
4. Crea una función PL/SQL que reciba tres números enteros (los tres lados del triángulo) y retorne el perímetro de dicho triángulo (suma de los lados).	8
5. Utiliza las dos funciones anteriores en una sentencia SELECT para obtener una relación de triángulos con el perímetro y el tipo.	9
Programación con PostgreSQL (5 puntos total)	10
6. Crea una base de datos llamada triángulos.	10
7. Crea un procedimiento almacenado dentro de la base de datos triángulos que realice las siguientes acciones:	11
a. Crea una tabla llamada triángulo con un único campo de tipo array donde se almacenarán los tres lados del triángulo. Si la tabla ya existe, se borrará y se volverá a crear de nuevo.....	11
b. Añade 20 filas a la tabla triángulo con valores al azar entre 1 y 5 para cada uno de los lados.	11
8. Crea una función PL/PGSQL que reciba un array (con los tres lados del triángulo) y retorne una cadena indicando si el triángulo es Equilátero, Isósceles o Escaleno.	15
6. Crea una función PL/PGSQL que reciba un array (con los tres lados del triángulo) y retorne el perímetro de dicho triángulo (suma de los lados).	17
7. Utiliza las dos funciones anteriores en una sentencia SELECT para obtener una relación de triángulos (cada lado en una columna independiente) con el perímetro y el tipo.	19
Enlace a GitHub	21
Bibliografía	22

Programación con MySQL (5 puntos en total)

1. Crea una base de datos llamada triángulos.



	#	Time	Action
✓	1	19:31:31	create database triángulos
✖	2	19:31:31	use triángulo
✓	3	19:31:48	use triángulos

2. Crea un procedimiento almacenado dentro de la base de datos triángulos que realice las siguientes acciones:

a. Crea una tabla llamada triángulo con los campos lado1, lado2 y lado3. Si la tabla ya existe, se borrará y se volverá a crear de nuevo.

b. Añade 20 filas a la tabla triangulo con valores al azar entre 1 y 5 para cada uno de los lados.

```
CREATE DEFINER='root'@'localhost' PROCEDURE  
'Crear_Triangulos'()
```

```
BEGIN
```

```
-- Primero declaro el i utilizaré en el while
```

```
    declare i int default 0;
```

```
-- Dropeo la tabla si existía ya la tabla
```

```
    drop table if exists triangulo;
```

```
-- Crear tabla triangulo con los atributos pertinentes
```

```
    create table triangulo(
```

```
        id_triangulo int auto_increment primary key,
```

```
        lado1 int,
```

```
        lado2 int,
```

```
        lado3 int
```

```
    );
```

```
-- Bucle while para sacar las 20 filas
```

```
    while i < 20 do
```

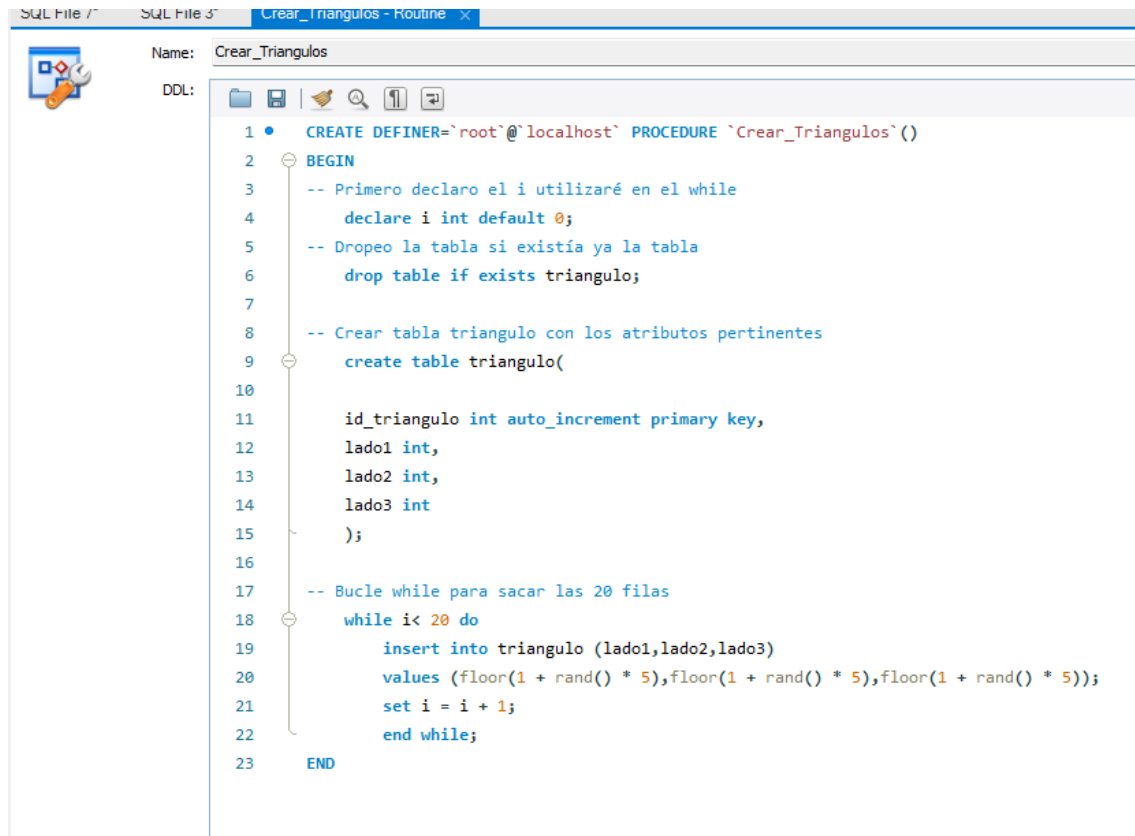
```
        insert into triangulo (lado1,lado2,lado3)
```

```
values (floor(1 + rand() * 5),floor(1 + rand() * 5),floor(1 +  
rand() * 5));
```

```
set i = i + 1;
```

```
end while;
```

```
END
```



The screenshot shows a MySQL IDE window titled 'Crear_Triangulos - Routine'. The 'DDL' tab is active, displaying a SQL script. The script is as follows:

```
1 • CREATE DEFINER=`root`@`localhost` PROCEDURE `Crear_Triangulos`()  
2 BEGIN  
3 -- Primero declaro el i utilizaré en el while  
4     declare i int default 0;  
5 -- Dropeo la tabla si existía ya la tabla  
6     drop table if exists triangulo;  
7  
8 -- Crear tabla triangulo con los atributos pertinentes  
9     create table triangulo(  
10  
11         id_triangulo int auto_increment primary key,  
12         lado1 int,  
13         lado2 int,  
14         lado3 int  
15     );  
16  
17 -- Bucle while para sacar las 20 filas  
18     while i < 20 do  
19         insert into triangulo (lado1,lado2,lado3)  
20         values (floor(1 + rand() * 5),floor(1 + rand() * 5),floor(1 + rand() * 5));  
21         set i = i + 1;  
22     end while;  
23 END
```

```

2
3 • use triángulos;
4
5 • select * from triangulo;
6 • call Crear_Triangulos();

```

Result Grid | Filter Rows: | Edit: | Export/Import:

	id_triángulo	lado1	lado2	lado3
▶	1	1	1	1
	2	3	1	1
	3	5	3	3
	4	1	5	3
	5	4	2	2
	6	4	2	4
	7	3	1	4
	8	5	2	3
	9	5	2	3
	10	2	2	2
	11	5	3	3
	12	1	3	4
	13	2	2	5
	14	1	5	1
	15	3	4	3
	16	5	3	3
	17	2	3	1
	18	2	2	1
	19	2	3	5
	20	4	1	1
*	NULL	NULL	NULL	NULL

3. Crea una función PL/SQL que reciba tres números enteros (los tres lados del triángulo) y retorna una cadena indicando si el triángulo es Equilátero, Isósceles o Escaleno.

- La función que he utilizado;

```

CREATE DEFINER='root'@'localhost' FUNCTION
`Tipo_Triangulo`(lado1 int,lado2 int,lado3 int) RETURNS
varchar(20) CHARSET utf8mb4
DETERMINISTIC

```

```
BEGIN

-- Declaro lo que quiero printear

declare Tipo_Triangulo varchar(20);

-- Establezco las condiciones para que sea un tipo de triángulo u
otro.

if lado1 = lado2 and lado2 = lado3 then
    set tipo_triangulo = "Es equilátero";
elseif lado1 = lado2 or lado1 = lado3 or lado2 = lado3 then
    set Tipo_Triangulo = "Es isósceles";
else
    set Tipo_Triangulo = "Es escaleno";
end if;

return Tipo_Triangulo;

END
```





```


8 • select id_triangulo, Tipo_Triangulo(lado1,lado2,lado3) from triangulo as Tipo_De_triangulo;
9
10 • select Perimetro (4,7,9);

```

result Grid

 Filter Rows:

Export: 

Wrap Cell Content: 

id_triangulo	Tipo_Triangulo(lado1,lado2,lado3)
1	Es equilátero
2	Es isósceles
3	Es isósceles
4	Es escaleno
5	Es isósceles
6	Es isósceles
7	Es escaleno
8	Es escaleno
9	Es escaleno
10	Es equilátero
11	Es isósceles
12	Es escaleno
13	Es isósceles
14	Es isósceles
15	Es isósceles
16	Es isósceles
17	Es escaleno
18	Es isósceles
19	Es escaleno
20	Es isósceles

4. Crea una función PL/SQL que reciba tres números enteros (los tres lados del triángulo) y retorne el perímetro de dicho triángulo (suma de los lados).

```

CREATE DEFINER=`root`@`localhost` FUNCTION
`Perimetro`(lado1 int,lado2 int, lado3 int ) RETURNS int
    DETERMINISTIC
BEGIN

RETURN lado1 + lado2 + lado3 ;

END

```

```
1 • CREATE DEFINER='root'@'localhost' FUNCTION `Perimetro` (lado1 int,lado2 int, lado3 int ) RETURNS int
2     DETERMINISTIC
3 BEGIN
4
5     RETURN lado1 + lado2 + lado3 ;
6 END
```

9

```
10 • select id_triangulo, Perimetro (lado1,lado2,lado3) from triangulo as Suma_Lados;
```

Result Grid		Filter Rows:	Export:	Wrap Cell Content:
	id_triangulo	Perimetro (lado1,lado2,lado3)		
▶	1	3		
	2	5		
	3	11		
	4	9		
	5	8		
	6	10		
	7	8		
	8	10		
	9	10		
	10	6		
	11	11		
	12	8		
	13	9		
	14	7		
	15	10		
	16	11		
	17	6		
	18	5		
	19	10		
	20	6		

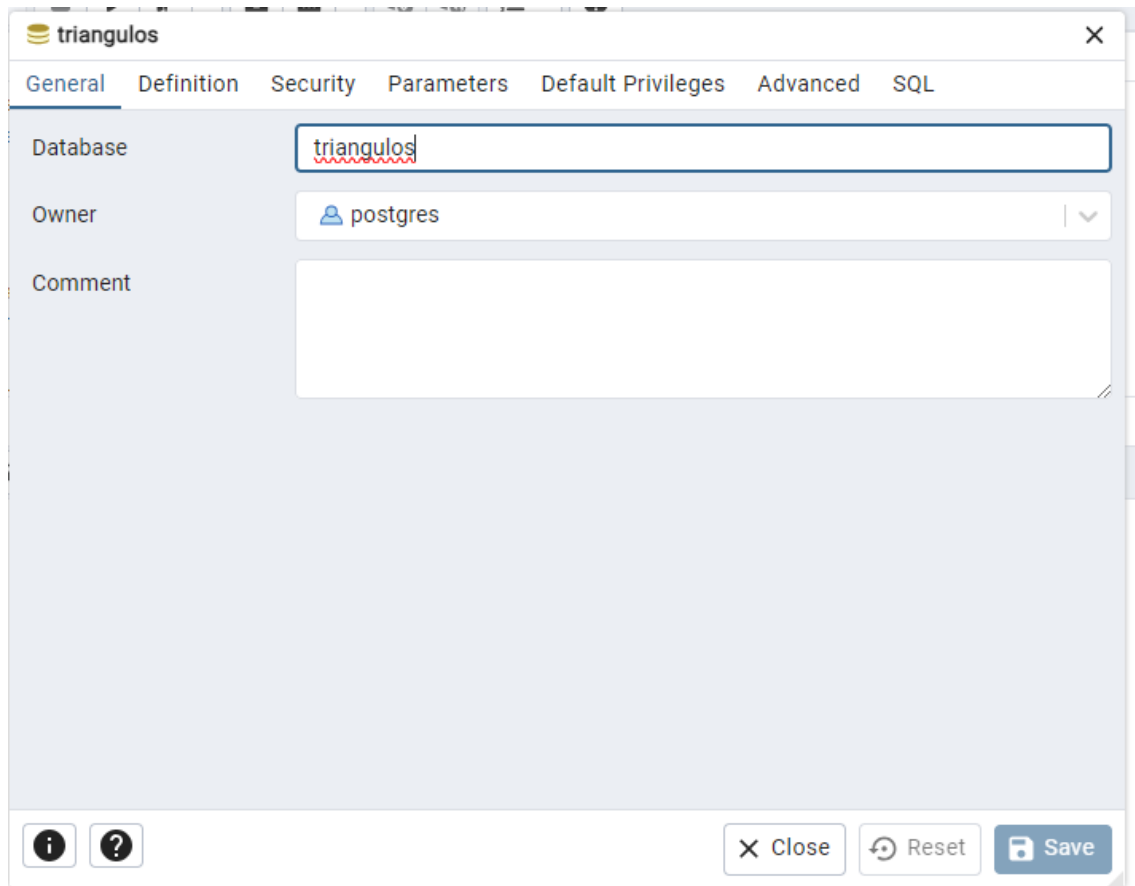
5. Utiliza las dos funciones anteriores en una sentencia SELECT para obtener una relación de triángulos con el perímetro y el tipo.

12 • `select id_triangulo, Tipo_Triangulo(lado1,lado2,lado3) as Tipo_De_triangulo , Perimetro (lado1,lado2,lado3) as Suma_Lados from triangulo;`

Result Grid			Filter Rows:	Export:	Wrap Cell Content:
id_triangulo	Tipo_De_triangulo	Suma_Lados			
1	Es equilátero	3			
2	Es isósceles	5			
3	Es isósceles	11			
4	Es escaleno	9			
5	Es isósceles	8			
6	Es isósceles	10			
7	Es escaleno	8			
8	Es escaleno	10			
9	Es escaleno	10			
10	Es equilátero	6			
11	Es isósceles	11			
12	Es escaleno	8			
13	Es isósceles	9			
14	Es isósceles	7			
15	Es isósceles	10			
16	Es isósceles	11			
17	Es escaleno	6			
18	Es isósceles	5			
19	Es escaleno	10			
20	Es isósceles	6			

Programación con PostgreSQL (5 puntos total)

6. Crea una base de datos llamada triángulos.



7. Crea un procedimiento almacenado dentro de la base de datos triángulos que realice las siguientes acciones:

- a. Crea una tabla llamada triángulo con un único campo de tipo array donde se almacenarán los tres lados del triángulo. Si la tabla ya existe, se borrará y se volverá a crear de nuevo.
- b. Añade 20 filas a la tabla triangulo con valores al azar entre 1 y 5 para cada uno de los lados.

-- Procedimiento base para crear la tabla y generar las 20 filas

CREATE OR REPLACE FUNCTION Crear_Triangulos()

```
RETURNS void AS $$
```

```
BEGIN
```

```
-- Eliminar la tabla si ya existe
```

```
DROP TABLE IF EXISTS triangulo;
```

```
-- Crear la tabla triangulo con un campo de tipo array
```

```
CREATE TABLE triangulo (
```

```
    id_triangulo SERIAL PRIMARY KEY,
```

```
    lados INT[]
```

```
);
```

```
-- Insertar 20 filas con valores aleatorios entre 1 y 5 para cada lado
```

```
FOR i IN 1..20 LOOP
```

```
    INSERT INTO triangulo (lados)
```

```
        VALUES (ARRAY[FLOOR(1 + RANDOM() * 5)::INT,  
FLOOR(1 + RANDOM() * 5)::INT, FLOOR(1 + RANDOM() *  
5)::INT]);
```

```
    END LOOP;
```

```
END;
```

```
$$ LANGUAGE plpgsql;
```

```
-- Procedimiento base para crear la tabla y generar las 20 filas
CREATE OR REPLACE FUNCTION Crear_Triangulos()

RETURNS void AS $$

BEGIN
    -- Eliminar la tabla si ya existe
    DROP TABLE IF EXISTS triangulo;

    -- Crear la tabla triangulo con un campo de tipo array
    CREATE TABLE triangulo (
        id_triangulo SERIAL PRIMARY KEY,
        lados INT[]

    );

    -- Insertar 20 filas con valores aleatorios entre 1 y 5 para cada lado
    FOR i IN 1..20 LOOP
        INSERT INTO triangulo (lados)
        VALUES (ARRAY[FLOOR(1 + RANDOM() * 5)::INT, FLOOR(1 + RANDOM() * 5)::INT, FLOOR(1 + RANDOM() * 5)::INT]);
    END LOOP;
END;
$$ LANGUAGE plpgsql;
```

```

25 select Crear_Triangulos()
26
27 select * from triangulo;
28

```

Data Output Messages Notifications



	lados integer[]
1	{1,3,3}
2	{5,1,5}
3	{3,4,2}
4	{5,4,1}
5	{2,5,3}
6	{2,1,4}
7	{3,2,3}
8	{5,1,4}
9	{1,1,3}
10	{5,5,4}
11	{4,1,3}
12	{5,2,1}
13	{1,4,3}
14	{2,1,4}
15	{2,4,1}
16	{3,4,3}
17	{3,5,1}
18	{1,5,1}
19	{5,2,5}
20	{2,5,5}

8. Crea una función PL/PGSQL que reciba un array (con los tres lados del triángulo) y retorne una cadena indicando si el triángulo es Equilátero, Isósceles o Escaleno.

```
-- Función para determinar el tipo de triángulo
CREATE OR REPLACE FUNCTION Tipo_Triangulo(lados INT[])
-- Que devuelva un varchar de 20
RETURNS VARCHAR(20) AS $$
-- Declaro el nombre de lo que deseo que devuelva
DECLARE
    Tipo_Triangulo VARCHAR(20);
-- Establezco las condiciones pertinentes para que funcione
correctamente
BEGIN
    IF lados[1] = lados[2] AND lados[2] = lados[3] THEN
        Tipo_Triangulo := 'Es equilátero';
    ELSIF lados[1] = lados[2] OR lados[1] = lados[3] OR lados[2] =
lados[3] THEN
        Tipo_Triangulo := 'Es isósceles';
    ELSE
        Tipo_Triangulo := 'Es escaleno';
    END IF;
    RETURN Tipo_Triangulo;
END;
$$ LANGUAGE plpgsql;
```



```

-- Función para determinar el tipo de triángulo
✓ CREATE OR REPLACE FUNCTION Tipo_Triangulo(lados INT[])
-- Que devuelva un varchar de 20
RETURNS VARCHAR(20) AS $$
-- Declaro el nombre de lo que deseo que devuelva
DECLARE
    Tipo_Triangulo VARCHAR(20);
-- Establezco las condiciones pertinentes para que funcione correctamente
✓ BEGIN
    IF lados[1] = lados[2] AND lados[2] = lados[3] THEN
        Tipo_Triangulo := 'Es equilátero';
    ✓ ELSIF lados[1] = lados[2] OR lados[1] = lados[3] OR lados[2] = lados[3] THEN
        Tipo_Triangulo := 'Es isósceles';
    ✓ ELSE
        Tipo_Triangulo := 'Es escaleno';
    END IF;
    RETURN Tipo_Triangulo;
END;
$$ LANGUAGE plpgsql;

```

```

50 select id_triángulo, Tipo_Triangulo(lados) as Tipo_de_triángulo from triangulo;
51
52
53
54

```

Data Output Messages Notifications



	id_triángulo [PK] integer	tipo_de_triángulo character varying
1	1	Es isósceles
2	2	Es isósceles
3	3	Es escaleno
4	4	Es escaleno
5	5	Es escaleno
6	6	Es escaleno
7	7	Es isósceles
8	8	Es escaleno
9	9	Es isósceles
10	10	Es escaleno
11	11	Es isósceles
12	12	Es isósceles
13	13	Es isósceles
14	14	Es escaleno
15	15	Es isósceles
16	16	Es escaleno
17	17	Es isósceles
18	18	Es escaleno
19	19	Es isósceles
20	20	Es escaleno

6. Crea una función PL/PGSQL que reciba un array (con los tres lados del triángulo) y retorne el perímetro de dicho triángulo (suma de los lados).

```
CREATE OR REPLACE FUNCTION Perimetro(lados INT[])  
RETURNS INT AS $$  
BEGIN  
    RETURN lados[1] + lados[2] + lados[3];  
END;  
$$ LANGUAGE plpgsql;
```

```

-- Función para calcular el perímetro de un triángulo
CREATE OR REPLACE FUNCTION Perimetro(lados INT[])
RETURNS INT AS $$
BEGIN
    RETURN lados[1] + lados[2] + lados[3];
END;
$$ LANGUAGE plpgsql;

61 -- Consulta para sacar los lados y la suma de todos ellos al final
62 SELECT lados[1] AS lado1, lados[2] AS lado2, lados[3] AS lado3,
63        Perimetro(lados) AS perimetro
64
65 FROM triangulo;
66
67
68
69
70

```

Data Output Messages Notifications

	lado1 integer	lado2 integer	lado3 integer	perimetro integer
1	4	3	4	11
2	4	2	4	10
3	3	5	1	9
4	5	2	3	10
5	5	2	1	8
6	5	1	2	8
7	3	1	3	7
8	3	2	1	6
9	3	1	1	5
10	4	1	2	7
11	2	1	1	4
12	4	4	2	10
13	4	4	2	10
14	1	3	5	9
15	3	3	1	7
16	2	1	5	8
17	3	4	4	11
18	2	4	3	9
19	5	2	5	12
20	2	1	4	7

7. Utiliza las dos funciones anteriores en una sentencia **SELECT** para obtener una relación de triángulos (cada lado en una columna independiente) con el perímetro y el tipo.

```
SELECT lados[1] AS lado1, lados[2] AS lado2, lados[3] AS lado3,  
       Perimetro(lados) AS perimetro,  
       Tipo_Triangulo(lados) AS tipo
```

```
FROM triangulo;
```

```
SELECT lados[1] AS lado1, lados[2] AS lado2, lados[3] AS lado3,  
       Perimetro(lados) AS perimetro,  
       Tipo_Triangulo(lados) AS tipo  
FROM triangulo;
```

```

70 SELECT lados[1] AS lado1, lados[2] AS lado2, lados[3] AS lado3,
71      Perimetro(lados) AS perimetro,
72      Tipo_Triangulo(lados) AS tipo
73
74 FROM triangulo;
75
76

```

Data Output Messages Notifications

	lado1 integer	lado2 integer	lado3 integer	perimetro integer	tipo character varying
1	4	3	4	11	Es isósceles
2	4	2	4	10	Es isósceles
3	3	5	1	9	Es escaleno
4	5	2	3	10	Es escaleno
5	5	2	1	8	Es escaleno
6	5	1	2	8	Es escaleno
7	3	1	3	7	Es isósceles
8	3	2	1	6	Es escaleno
9	3	1	1	5	Es isósceles
10	4	1	2	7	Es escaleno
11	2	1	1	4	Es isósceles
12	4	4	2	10	Es isósceles
13	4	4	2	10	Es isósceles
14	1	3	5	9	Es escaleno
15	3	3	1	7	Es isósceles
16	2	1	5	8	Es escaleno
17	3	4	4	11	Es isósceles
18	2	4	3	9	Es escaleno
19	5	2	5	12	Es isósceles
20	2	1	4	7	Es escaleno
Total rows: 20		Query complete 00:00:00.081			

Enlace a GitHub

<https://github.com/Cortes-cmd/BBDD.git>

Bibliografía

Arrays en Postgres · Etaoin Shrldu. (s/f). Netlify.app. Recuperado el 10 de febrero de 2025, de <https://etaoinshrldu.netlify.app/arrays-en-postgres/>

ChatGPT. (s/f). Chatgpt.com. Recuperado el 10 de febrero de 2025, de <https://chatgpt.com/c/67a9bab2-ff40-8001-b47e-af98967ac466>

Como utilizar un array dinámico en función plpgsql. (s/f). Narkive.com. Recuperado el 10 de febrero de 2025, de <https://pgsql-es-ayuda.postgresql.narkive.com/m3RoLpkh/como-utilizar-un-array-dinamico-en-funcion-plpgsql>

Grupo de sierras perforadoras de Beijing. (2023, agosto 31). Uso de matrices en PostgreSQL: una guía. es.xtshengguo.com. <https://es.xtshengguo.com/news/using-arrays-in-postgresql-a-guide.html>

Hosting, S. W. (2020, enero 7). Tutorial de MySQL Workbench. SiteGround. <https://www.siteground.es/tutoriales/php-mysql/mysql-workbench/>

Manejo de arrays en PostgreSQL - Foros Club Delphi. (s/f). Clubdelphi.com. Recuperado el 10 de febrero de 2025, de <https://www.clubdelphi.com/foros/showthread.php?t=91918>

pasar un array a una funcion en postgresql. (s/f). Stack Overflow en español. Recuperado el 10 de febrero de 2025, de <https://es.stackoverflow.com/questions/260042/pasar-un-array-a-una-funcion-en-postgresql>