

PRÁCTICA 4.3

Automatización. PostgreSQL.

DATOS:

- **Ciclo formativo:** Grado Superior 2º ASIR A
- **Módulo:** Administración de Sistemas Gestores de Bases de Datos
- **Unidad de trabajo:** UD04 – Configuración de una SGBD
- **Nombre y apellidos:** MC Pareja Ferreira

ÍNDICE

INTRODUCCIÓN.....	3
APARTADO 1. Rutinas almacenadas.....	4
APARTADO 2. Eventos.	5
APARTADO 3. Triggers.	7
APARTADO 4. Vistas.	8

INTRODUCCIÓN

Para este ejercicio trabajaremos con estas 3 tablas: clientes, ventas y vip.

Query Query History Messages Notifications

```

1 SELECT * FROM public.clientes
2 ORDER BY id_cliente ASC
    
```

Data Output

	id_cliente [PK] integer	nombre character varying (50)	apellido character varying (50)	email character varying (100)	telefono character varying (15)	antiguedad_en_semanas integer
1	1	Juan	Pérez	juanperez@gmail.com	123456789	3
2	2	María	González	mariagonzalez@hotmail.com	987654321	6
3	3	Pedro	Ramírez	pedroramirez@yahoo.com	456123789	2
4	4	Lucía	García	luciagarcia@gmail.com	789123456	8

public.ventas/entrega3/postgres@SGBD

No limit

Query Query History Messages Notifications

```

1 SELECT * FROM public.ventas
2 ORDER BY id_venta ASC

```

Data Output

	id_venta [PK] integer	id_cliente integer	precio_total numeric (10,2)	para_regalo boolean	fecha_venta date
1	1	1	50.00	false	2023-02-01
2	2	1	100.00	true	2023-02-10
3	3	2	50.00	true	2023-02-15
4	4	3	75.00	true	2023-02-20
5	5	4	30.00	false	2023-02-08
6	6	4	150.00	false	2023-02-19

No limit

Query

Query History

Messages

Notifications

```

1 SELECT * FROM public.vip
2 ORDER BY id_cliente ASC

```

Data Output

	id_cliente [PK] integer	nombre character varying (50)	facturacion numeric (10,2)
1	4	Lucía García	180.00

APARTADO 1. Rutinas almacenadas.

Define un procedimiento que sume 1 a todas las filas que cumplan X condición.

En mi caso, aumentaremos en 1€ el precio de la venta en aquellas compras que hayan sido marcadas como regalo para compensar el gasto del envoltorio.

```
entrega3/postgres@SGBD
Query Query History Messages Notifications
1 CREATE OR REPLACE PROCEDURE actualizar_precio_venta()
2 LANGUAGE SQL
3 AS $$
4     UPDATE ventas
5     SET precio_total = precio_total + 1
6     WHERE para_regalo = true;
7 $$;
8
9 CALL actualizar_precio_venta();
10
```

Tras llamar al procedimiento, así es como queda la tabla:

```
public.ventas/entrega3/postgres@SGBD
Query Query History Messages Notifications
1 SELECT * FROM public.ventas
2 ORDER BY id_venta ASC
3
```

Data Output

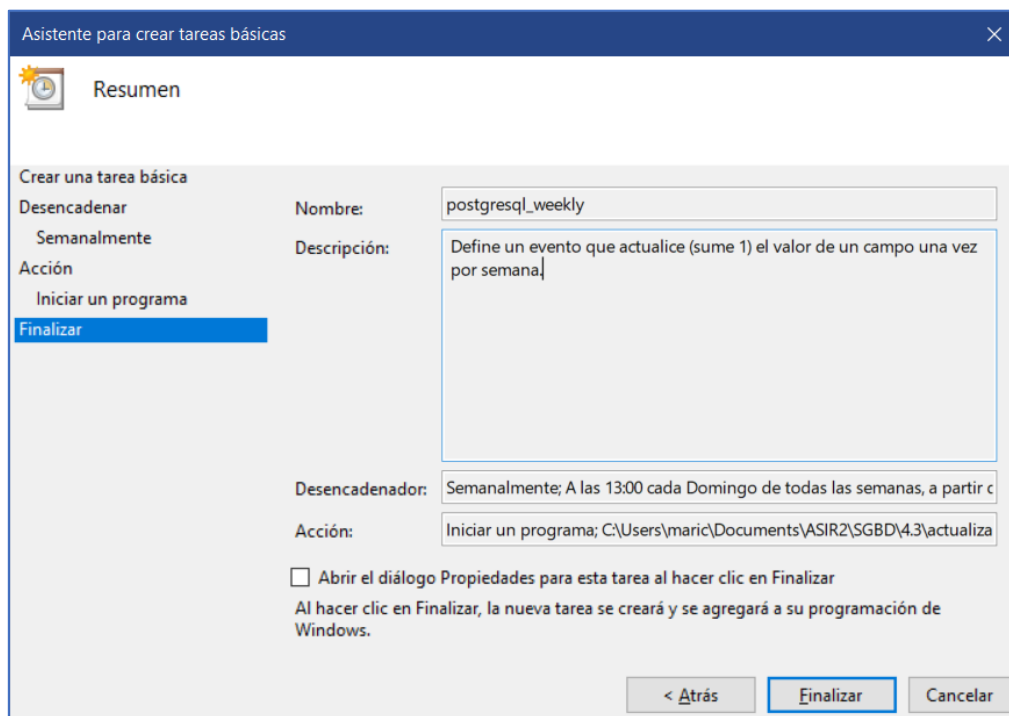
	id_venta [PK] integer	id_cliente integer	precio_total numeric (10,2)	para_regalo boolean	fecha_venta date
1	1	1	50.00	false	2023-02-01
2	2	1	101.00	true	2023-02-10
3	3	2	51.00	true	2023-02-15
4	4	3	76.00	true	2023-02-20
5	5	4	30.00	false	2023-02-08
6	6	4	150.00	false	2023-02-19

APARTADO 2. Eventos.

Definir un evento que actualice (sume 1) el valor de un campo una vez por semana.

Para este ejercicio, aumentaremos de forma semanal en 1 la “antigüedad_en_semanas” del cliente. Sin embargo, PostgreSQL no tiene un concepto de eventos como pueden tener otros sistemas de bases de datos y no es posible realizar esta actividad sin utilizar extensiones como *pg_cron* (para Linux) o *pgAgent* (versión Enterprise).

Como estoy desarrollando la actividad en Windows, haré uso del programador de tareas de Windows para ejecutar un punto .bat de forma repetitiva una vez a la semana.



Aquí el contenido del .bat y del script .sql:

```
actualizar_antiguedad_semanalmente.bat: Bloc de notas
Archivo Edición Formato Ver Ayuda
@echo off
SET PGPASSWORD=
psql -U postgres -d entrega3 -f actualizar_antiguedad_semanalmente.sql
```

```
actualizar_antiguedad_semanalmente.sql: Bloc de notas
Archivo Edición Formato Ver Ayuda
UPDATE clientes SET antiguedad_en_semanas = antiguedad_en_semanas + 1;
```

Línea 1, columna 1 100% Windows (CRLF) UTF-8

Si intentamos ejecutar el .bat para comprobar su funcionamiento, veremos como se aplican los cambios.

public.clientes/entrega3/postgres@SGBD

No limit

```

1 SELECT * FROM public.clientes
2 ORDER BY id_cliente ASC

```

Data Output

	id_cliente [PK] integer	nombre character varying (50)	apellido character varying (50)	email character varying (100)	telefono character varying (15)	antiguedad_en_semanas integer
1	1	Juan	Pérez	juanperez@gmail.com	123456789	5
2	2	Maria	González	mariagonzalez@hotmail.com	987654321	7
3	3	Pedro	Ramírez	pedroramirez@yahoo.com	456123789	3
4	4	Lucía	García	luciagarcia@gmail.com	789123456	9

APARTADO 3. Triggers.

Define un trigger que cree una entrada en la tabla X cada vez que se inserte en la tabla Y una fila que sobrepase un límite.

En relación a mis tablas, cada vez que se inserte o actualice una venta, si la facturación total del cliente es mayor a 100 y este supera las 5 semanas de antigüedad, se insertará su id_cliente, su nombre y su facturación total en la tabla VIP. En el caso de que el cliente ya existiera en la tabla, solo se actualizará su facturación.

```
entrega3/postgres@SGBD
Query Query History Messages Notifications
1 CREATE OR REPLACE FUNCTION actualizar_vip() RETURNS TRIGGER AS $$
2 DECLARE
3     v_facturacion DECIMAL(10,2);
4     v_antiguedad INTEGER;
5     v_nombre VARCHAR(50);
6 BEGIN
7     SELECT SUM(precio_total) INTO v_facturacion FROM ventas WHERE id_cliente = NEW.id_cliente;
8     SELECT antiguedad_en_semanas INTO v_antiguedad FROM clientes WHERE id_cliente = NEW.id_cliente;
9     SELECT nombre INTO v_nombre FROM clientes WHERE id_cliente = NEW.id_cliente;
10
11 IF (v_facturacion > 100 AND v_antiguedad >=5) THEN
12 IF EXISTS (SELECT 1 FROM VIP WHERE id_cliente = NEW.id_cliente) THEN
13     UPDATE VIP SET facturacion = v_facturacion
14     WHERE id_cliente = NEW.id_cliente;
15 ELSE
16     INSERT INTO VIP (id_cliente, nombre, facturacion)
17     VALUES (NEW.id_cliente, v_nombre, v_facturacion);
18 END IF;
19 END IF;
20
21 RETURN NEW;
22 END;
23 $$ LANGUAGE plpgsql;
24
25 CREATE TRIGGER actualizar_vip_trigger
26 AFTER INSERT OR UPDATE ON ventas
27 FOR EACH ROW
28 EXECUTE FUNCTION actualizar_vip();
```

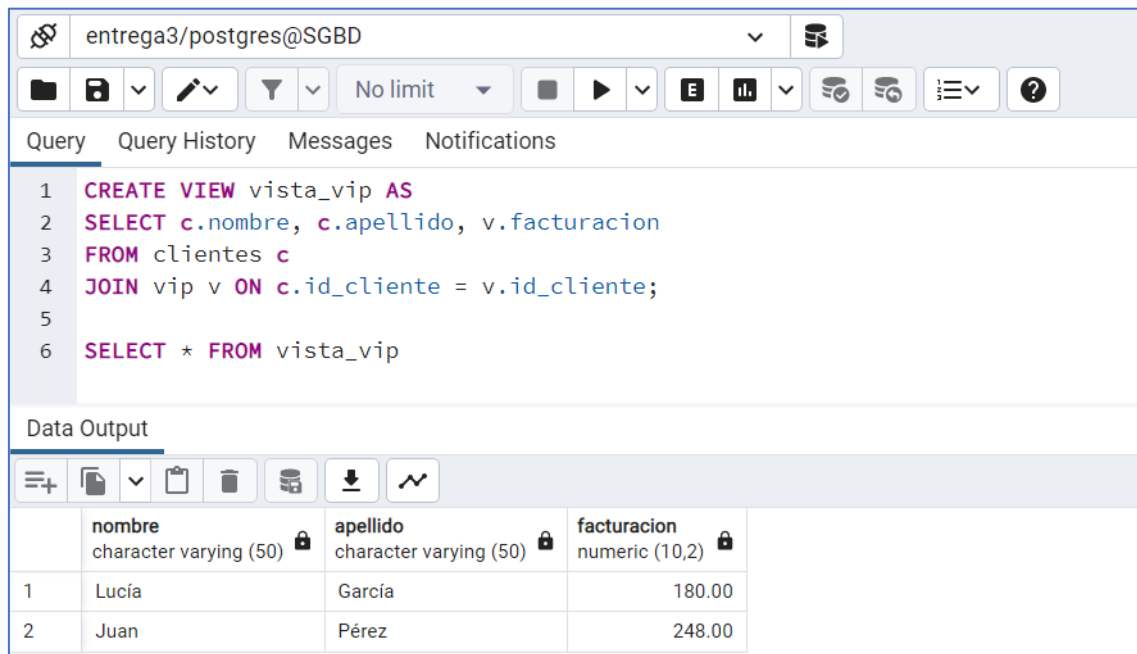
Si creamos una nueva venta para el id_cliente 1, que en el anterior ejercicio ya cumplió las 5 semanas de antigüedad, y cuyos gastos son mayores a 100€ nos insertará una nueva fila en la tabla VIP.

public.vip/entrega3/postgres@SGBD			
Query Query History Messages Notifications			
1 SELECT * FROM public.vip			
2 ORDER BY id_cliente ASC			
Data Output			
	id_cliente [PK] integer	nombre character varying (50)	facturacion numeric (10,2)
1	1	Juan	248.00
2	4	Lucía García	180.00

APARTADO 4. Vistas.

Crea una vista que muestre el nombre, apellido y facturación de la tabla Clientes con la opción de seguridad de la persona que la invoca.

Creemos la vista y mostramos el resultado de la misma.



The screenshot shows a PostgreSQL client interface with the username 'entrega3/postgres@SGBD'. The query editor contains the following SQL code:

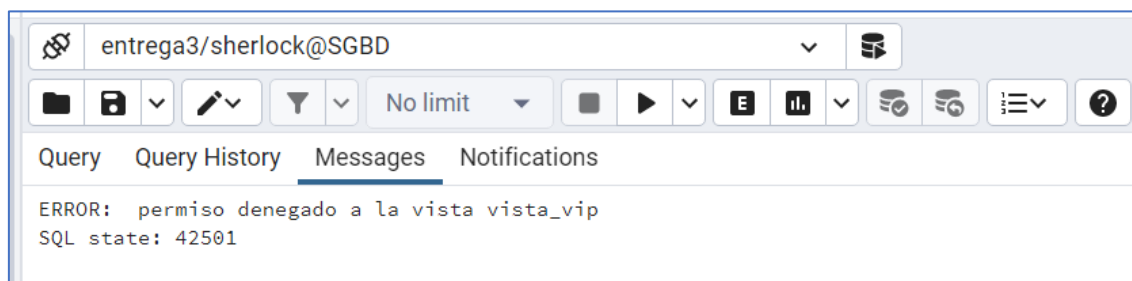
```
1 CREATE VIEW vista_vip AS
2 SELECT c.nombre, c.apellido, v.facturacion
3 FROM clientes c
4 JOIN vip v ON c.id_cliente = v.id_cliente;
5
6 SELECT * FROM vista_vip
```

The 'Data Output' tab is active, displaying the results of the query in a table:

	nombre character varying (50)	apellido character varying (50)	facturacion numeric (10,2)
1	Lucía	García	180.00
2	Juan	Pérez	248.00

Después intenta acceder a esta vista con un usuario que no tenga los mismos permisos. ¿Qué ocurriría? ¿Cómo podemos cambiar esta situación y que el segundo usuario pueda utilizar la vista como el primero?

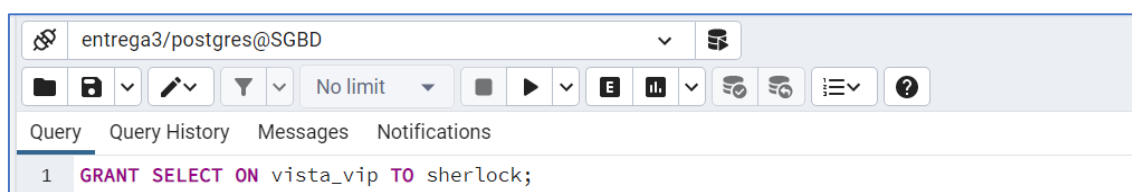
Si intentamos acceder a la vista con otro usuario que no posea los permisos necesarios, nos aparecerá este mensaje:



The screenshot shows the same PostgreSQL client interface, but the 'Messages' tab is active. It displays the following error message:

```
ERROR: permiso denegado a la vista vista_vip
SQL state: 42501
```

Para que este pueda acceder, el creador de la base de datos deberá concederle permisos de lectura sobre la vista.



The screenshot shows the PostgreSQL client interface with the username 'entrega3/postgres@SGBD'. The query editor contains the following SQL code:

```
1 GRANT SELECT ON vista_vip TO sherlock;
```


Ahora el usuario “sherlock” podrá leer la vista.

entrega3/sherlock@SGBD

Query Query History Messages Notifications

```
1 SELECT * FROM vista_vip
```

Data Output

	nombre character varying (50) 🔒	apellido character varying (50) 🔒	facturacion numeric (10,2) 🔒
1	Lucía	García	180.00
2	Juan	Pérez	248.00