

Base de Datos I

Dr. Edward Hinojosa Cárdenas
ehinojosa@unsa.edu.pe

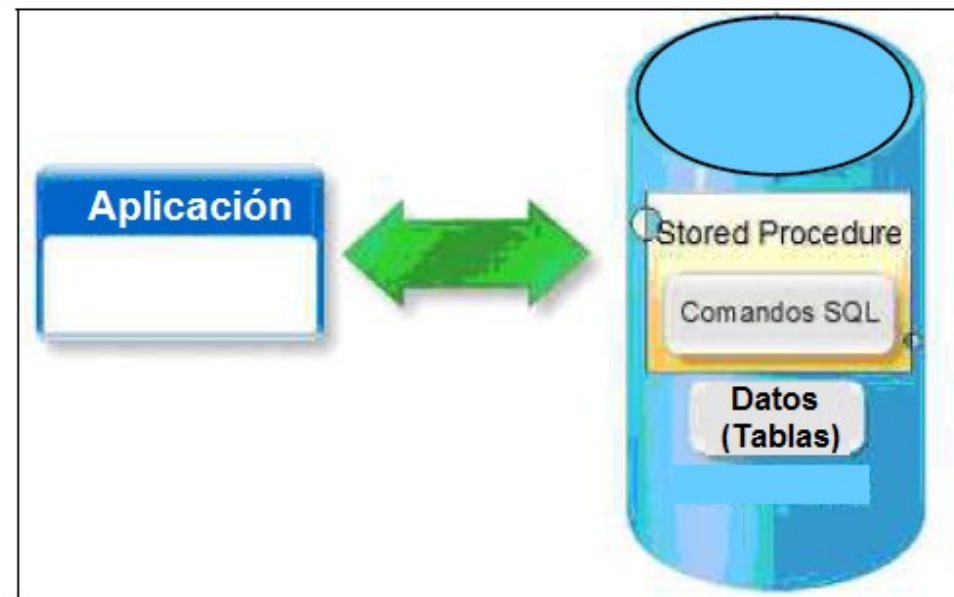
Procedimientos Almacenados (Stored Procedures)

- SP es un conjunto de comandos al cual es atribuido un nombre.
- Este conjunto se encuentra almacenado en la BD y puede ser invocado en cualquier momento por el SGBD o por un sistema que utiliza la misma.

Razones para usar SP

- SP son utilizados para mover gran parte de código de manipulación de datos para el servidor, ello elimina la transferencia de datos del servidor al cliente, por la red, para manipulación → reducción de tráfico de la red → mejor performance general.

Razones para usar SP



Razones para usar SP

- Siempre que una aplicación cliente envía un comando SQL para el servidor, el comando tiene que tener la sintaxis analizada y, a continuación, es enviado a un programa optimizador del SGBD para la formulación de un plan de ejecución.
- Los SP son analizados y optimizados una única vez (cuando son creados), presentan mejor rendimiento que los comandos SQL enviados por las aplicaciones.

Razones para usar SP

- SP pueden reducir el tiempo de desarrollo y manutención de los sistemas pues las SP pueden ser compartidos por todas las aplicaciones existentes. El mantenimiento es sencillo porque es posible alterar un SP sin tener que alterar y/o recompilar cada aplicación cliente.
- Puede ser más veloz dado que generalmente el servidor es una de las máquinas más poderosas en la red.

Sintaxis SP

```
CREATE [ OR REPLACE ] PROCEDURE
    name ( [ [ argmode ] [ argname ] argtype [ { DEFAULT | = } default_expr ] [, ...] ] )
{ LANGUAGE lang_name
  | TRANSFORM { FOR TYPE type_name } [, ... ]
  | [ EXTERNAL ] SECURITY INVOKER | [ EXTERNAL ] SECURITY DEFINER
  | SET configuration_parameter { TO value | = value | FROM CURRENT }
  | AS 'definition'
  | AS 'obj_file', 'link_symbol'
} ...
```

```
DROP PROCEDURE [ IF EXISTS ] name [ ( [ [ argmode ] [ argname ] argtype [, ...] ] ) ] [, ...]
    [ CASCADE | RESTRICT ]
```

Ejemplo 1

The screenshot shows a web-based PostgreSQL query editor interface. At the top, there is a navigation bar with tabs: Dashboard, Properties, SQL, Statistics, Dependencies, and Dependents. The current tab is 'SQL', and the connection is 'postgres/postgres@localhost *'. Below the navigation bar is a toolbar with various icons for file operations, search, and execution. The main area is divided into two sections: 'Query Editor' and 'Query History'. The 'Query Editor' contains a single query: `1 SELECT VERSION();`. Below the query editor, there are tabs for 'Data Output', 'Explain', 'Messages', and 'Notifications'. The 'Data Output' tab is active, showing a table with one row of results. The table has a column named 'version' of type 'text'. The result text is: 'PostgreSQL 11.5 (Ubuntu 11.5-3.pgdg18.04+1) on x86_64-pc-linux-gnu, compiled by gcc (Ubuntu 7.4.0-1ubuntu1~18.04.1) 7.4.0, 64-bit'.

Dashboard Properties SQL Statistics Dependencies Dependents postgres/postgres@localhost *

postgres/postgres@localhost

Query Editor Query History

1 `SELECT VERSION();`

Data Output Explain Messages Notifications

	version text	
1	PostgreSQL 11.5 (Ubuntu 11.5-3.pgdg18.04+1) on x86_64-pc-linux-gnu, compiled by gcc (Ubuntu 7.4.0-1ubuntu1~18.04.1) 7.4.0, 64-bit	

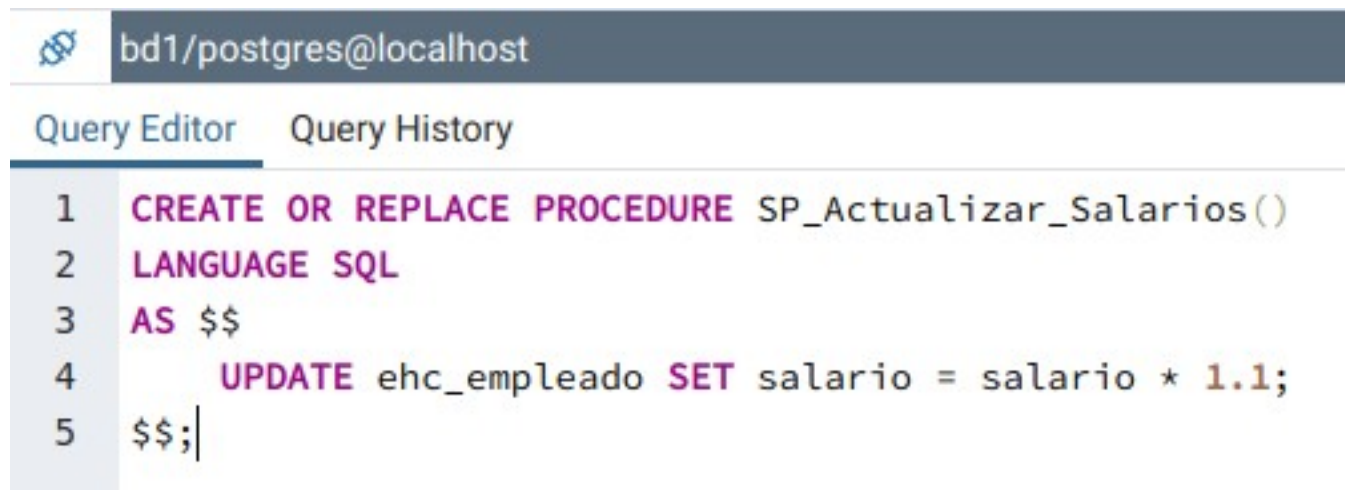
Ejemplo 1

- Aumentar el salario de todos los empleados en un 10% usando un SP:

	dni [PK] integer	nombres character varying (100)	papellido character varying (100)	sapellido character varying (100)	direccion character varying (150)	fecha_nacimiento date	sexo character (1)	salario numeric (10,2)	super_dni integer	dcodigo integer
1	888665555	Francisco	Linares	Gomez	Calle Numero H 8	1957-11-10	M	5500.00	[null]	1
2	123456789	Juan	Perez	Rodriguez	Calle Numero A 1	1965-01-09	M	300.00	333445555	5
3	333445555	Frank	Velazquez	Flores	Calle Numero B 2	1955-12-08	M	4000.00	888665555	5
4	999887777	Alice	Jimenez	Portugal	Calle Numero C 3	1968-07-19	F	2500.00	987654321	4
5	987654321	Luisa	Santos	Ferrel	Calle Numero D 4	1951-06-20	F	430.00	888665555	4
6	666884444	Pedro	Lima	Maldonado	Calle Numero E 5	1952-09-15	M	1200.00	333445555	5
7	453453453	Daniela	Acco	Olvarez	Calle Numero F 6	1962-07-31	F	2500.00	333445555	5
8	987987987	Mateo	Vela	Marruecos	Calle Numero G 7	1979-03-29	M	2500.00	987654321	4

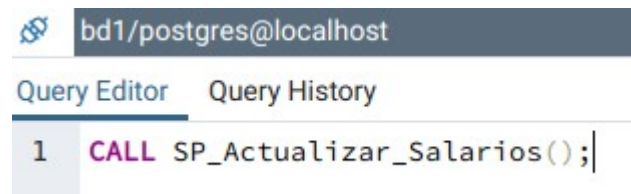
Ejemplo 1

- Aumentar el salario de todos los empleados en un 10% usando un SP:



The screenshot shows a PostgreSQL Query Editor window with the connection 'bd1/postgres@localhost'. The 'Query Editor' tab is active. The SQL code is as follows:

```
1 CREATE OR REPLACE PROCEDURE SP_Actualizar_Salarios()  
2 LANGUAGE SQL  
3 AS $$  
4     UPDATE ehc_empleado SET salario = salario * 1.1;  
5 $$;
```



The screenshot shows a PostgreSQL Query Editor window with the connection 'bd1/postgres@localhost'. The 'Query Editor' tab is active. The SQL code is as follows:

```
1 CALL SP_Actualizar_Salarios();
```

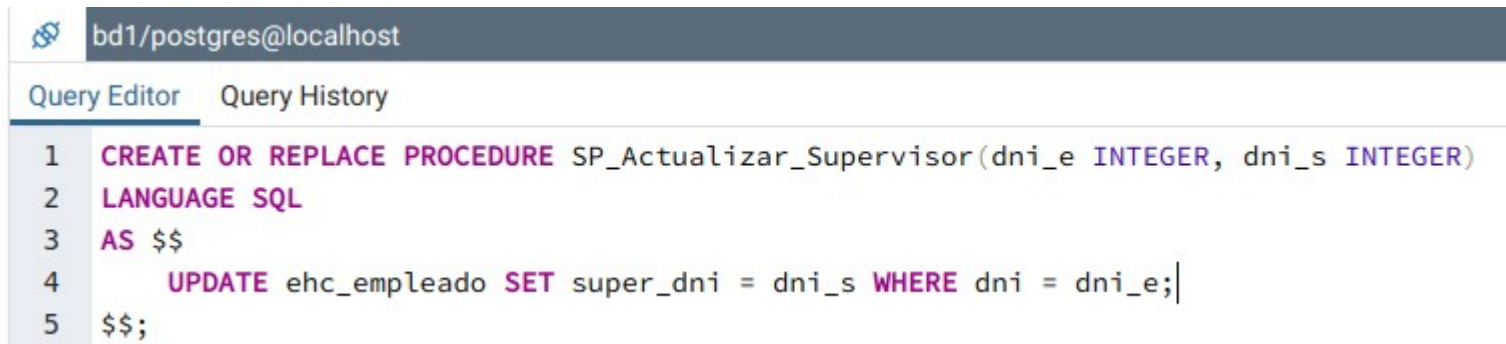
Ejemplo 1

- Y obtenemos:

Data Output		Explain	Messages	Notifications						
	dni [PK] integer	nombres character varying (100)	apellido character varying (100)	sapellido character varying (100)	direccion character varying (150)	fecha_nacimiento date	sexo character (1)	salario numeric (10,2)	super_dni integer	dcodigo integer
1	888665555	Francisco	Linares	Gomez	Calle Numero H 8	1957-11-10	M	6050.00	[null]	1
2	123456789	Juan	Perez	Rodriguez	Calle Numero A 1	1965-01-09	M	330.00	333445555	5
3	333445555	Frank	Velazquez	Flores	Calle Numero B 2	1955-12-08	M	4400.00	888665555	5
4	999887777	Alice	Jimenez	Portugal	Calle Numero C 3	1968-07-19	F	2750.00	987654321	4
5	987654321	Luisa	Santos	Ferrel	Calle Numero D 4	1951-06-20	F	473.00	888665555	4
6	666884444	Pedro	Lima	Maldonado	Calle Numero E 5	1952-09-15	M	1320.00	333445555	5
7	453453453	Daniela	Acco	Olvarez	Calle Numero F 6	1962-07-31	F	2750.00	333445555	5
8	987987987	Mateo	Vela	Marruecos	Calle Numero G 7	1979-03-29	M	2750.00	987654321	4

Ejemplo 2

- Modificar el supervisor de un determinado empleado usando un SP (recibe como parámetro el código del empleado y código del supervisor):



The screenshot shows a PostgreSQL Query Editor window with the connection 'bd1/postgres@localhost'. The 'Query Editor' tab is active, displaying the following SQL code:

```
1 CREATE OR REPLACE PROCEDURE SP_Actualizar_Supervisor(dni_e INTEGER, dni_s INTEGER)
2 LANGUAGE SQL
3 AS $$
4     UPDATE ehc_empleado SET super_dni = dni_s WHERE dni = dni_e;
5 $$;
```

Ejemplo 2



The screenshot shows a PostgreSQL query editor window. At the top, there is a dark blue header bar with a PostgreSQL logo icon on the left and the text "bd1/postgres@localhost" on the right. Below the header, there are two tabs: "Query Editor" and "Query History". The "Query Editor" tab is currently selected and highlighted with a blue underline. Below the tabs, the query text is displayed on a light gray background. The first line of the query is numbered "1" in a light gray box, followed by the text "CALL SP_Actualizar_Supervisor(888665555, 987654321);". The word "CALL" is highlighted in purple, and the numbers "888665555" and "987654321" are highlighted in brown.

```
bd1/postgres@localhost
```

Query Editor Query History

```
1 CALL SP_Actualizar_Supervisor(888665555, 987654321);
```

Ejemplo 2

Data Output Explain Messages Notifications

	dni [PK] integer	nombres character varying (100)	papellido character varying (100)	sapellido character varying (100)	direccion character varying (150)	fecha_nacimiento date	sexo character (1)	salario numeric (10,2)	super_dni integer	dcodigo integer
1	123456789	Juan	Perez	Rodriguez	Calle Numero A 1	1965-01-09	M	330.00	333445555	5
2	333445555	Frank	Velazquez	Flores	Calle Numero B 2	1955-12-08	M	4400.00	888665555	5
3	999887777	Alice	Jimenez	Portugal	Calle Numero C 3	1968-07-19	F	2750.00	987654321	4
4	987654321	Luisa	Santos	Ferrel	Calle Numero D 4	1951-06-20	F	473.00	888665555	4
5	666884444	Pedro	Lima	Maldonado	Calle Numero E 5	1952-09-15	M	1320.00	333445555	5
6	453453453	Daniela	Acco	Olvarez	Calle Numero F 6	1962-07-31	F	2750.00	333445555	5
7	987987987	Mateo	Vela	Marruecos	Calle Numero G 7	1979-03-29	M	2750.00	987654321	4
8	888665555	Francisco	Linares	Gomez	Calle Numero H 8	1957-11-10	M	6050.00	987654321	1

Funciones

- Al igual que los SP, tener nuestras funciones en lugar de procesar los datos con algún lenguaje del lado del servidor, como PHP , tiene la ventaja de que se transfiere menos información de la BD al servidor web.
- Con el consiguiente aumento del rendimiento y que estas funciones harán que podamos atacar la base de datos desde cualquier otro lenguaje, como Java o ASP.NET sin tener que volver a procesar los datos otra vez.

Funciones

- PostgreSQL tiene muchas funciones que podemos usar en nuestro procedimientos almacenados y consultas, pero en ocasiones podemos necesitar crear nuestras propias funciones para hacer cosas más especializadas.
- Vamos a ver cómo crear funciones en PostgreSQL.

Funciones

```
CREATE [ OR REPLACE ] FUNCTION
    name ( [ [ argmode ] [ argname ] argtype [ { DEFAULT | = } default_expr ] [, ...] ] )
    [ RETURNS rettype
      | RETURNS TABLE ( column_name column_type [, ...] ) ]
{ LANGUAGE lang_name
  | TRANSFORM { FOR TYPE type_name } [, ... ]
  | WINDOW
  | IMMUTABLE | STABLE | VOLATILE | [ NOT ] LEAKPROOF
  | CALLED ON NULL INPUT | RETURNS NULL ON NULL INPUT | STRICT
  | [ EXTERNAL ] SECURITY INVOKER | [ EXTERNAL ] SECURITY DEFINER
  | PARALLEL { UNSAFE | RESTRICTED | SAFE }
  | COST execution_cost
  | ROWS result_rows
  | SET configuration_parameter { TO value | = value | FROM CURRENT }
  | AS 'definition'
  | AS 'obj_file', 'link_symbol'
} ...
```

Ejemplo 1

- Mostrar el DNI, nombre, primer apellido y fecha de nacimiento de todos los empleados que se encuentren entre las fechas de inicio y fin enviados como parámetros usando una función:

bd1/postgres@localhost

Query Editor Query History

```
1 CREATE FUNCTION FU_Empleados_Filtro_Fechas(fecha_ini DATE, fecha_fin DATE)
2 RETURNS TABLE(dni INTEGER, nombre VARCHAR(50), papellido VARCHAR(100), fecha_nacimiento DATE) AS $$
3     SELECT dni, nombres, papellido, fecha_nacimiento FROM ehc_empleado WHERE fecha_nacimiento BETWEEN fecha_ini AND fecha_fin;
4 $$
5 LANGUAGE SQL;
```

Ejemplo 1

bd1/postgres@localhost

Query Editor Query History

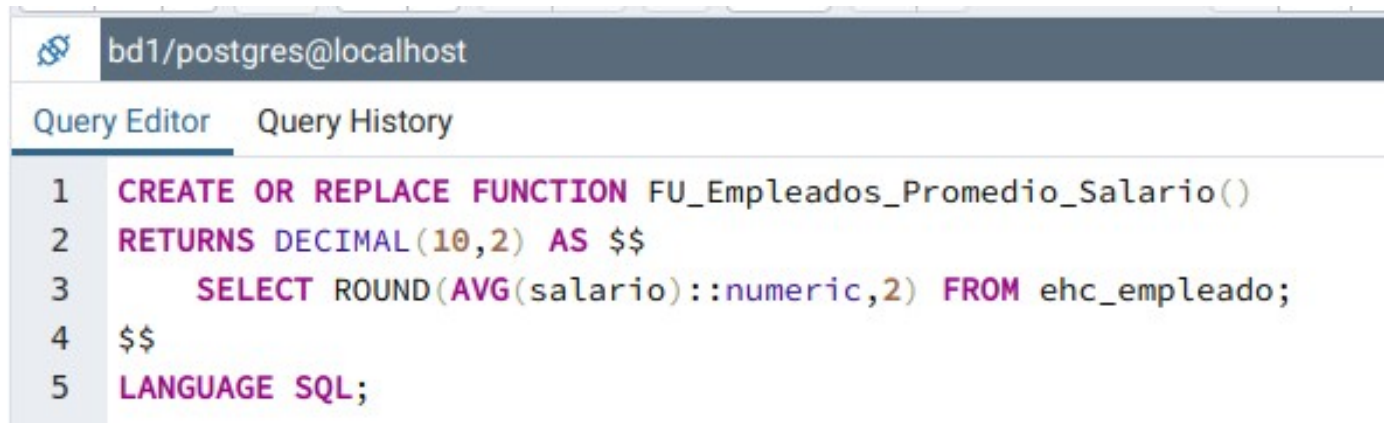
```
1 SELECT FU_Empleados_Filtro_Fechas('1965-01-01', '1969-12-31');
```

Data Output Explain Messages Notifications

	fu_empleados_filtro_fechas record	
1	(123456789,Juan,Perez,1965-01-09)	
2	(999887777,Alice,Jimenez,1968-07-19)	

Ejemplo 2

- Mostrar el promedio de salario de todos los empleados usando una función:



```
bd1/postgres@localhost  
Query Editor  Query History  
1  CREATE OR REPLACE FUNCTION FU_Empleados_Promedio_Salario()  
2  RETURNS DECIMAL(10,2) AS $$  
3      SELECT ROUND(AVG(salario)::numeric,2) FROM ehc_empleado;  
4  $$  
5  LANGUAGE SQL;
```

Ejemplo 2

bd1/postgres@localhost

Query Editor Query History

```
1 SELECT FU_Empleados_Promedio_Salario();
```

Data Output Explain Messages Notifications

	fu_empleados_promedio_salario	
	numeric	
1	2602.88	

GRACIAS

Dr. Edward Hinojosa Cárdenas
ehinojosa@unsa.edu.pe