

Clausula SELECT de PostgreSQL

La clausula `SELECT` es una de las más utilizadas cuando se trabaja con datos. Esta permite extraer datos de una o varias tablas. Para ello la clausula `SELECT` puede enriquecerse con otras clausulas que nos permitirán filtrar los datos hasta conseguir el conjunto exacto que necesitamos. Por todo ello, podemos decir que la clausula `SELECT` es una de las más complejas que existen en PostgreSQL.

Dado que la clausula `SELECT` es una clausula compleja, iremos presentando sus diferentes componentes en diversos capítulos. En este primer capítulo nos centraremos en la sintaxis básica de la clausula `SELECT`.

Sintaxis

La sintaxis básica de la clausula `SELECT` es la siguiente:

```
SELECT
  columnas
FROM
  tabla;
```

Como ves la sentencia `SELECT` tiene una sintaxis muy intuitiva que puede ser traducida fácilmente al lenguaje natural: **selecciona** la lista de **columnas de la tabla**. Ahora veamos con más detalle los componentes:

- `SELECT`. Es el nombre de la clausula e indica al motor de bases de datos que deseamos ejecutar una consulta.
- `columnas`. Es una lista de todas columnas que queremos recuperar de la tabla `tabla`. La lista de columnas puede ser uno o varios nombres de columna separados por una coma (,). También podemos usar el carácter asterisco (*), que significa 'todas' y nos permite seleccionar todas las columnas sin tener que definir los nombres.
- `FROM`. Es una clausula dentro de `SELECT` que define una lista de las tablas desde donde queremos recuperar datos. Esta clausula es opcional y puede ser omitida cuando no se van a recuperar datos de ninguna tabla.
- `tabla`. Es la lista propiamente dicha de tablas sobre las cuales queremos recuperar datos. La lista puede ser de una o más tablas separadas por una coma.

Recuerda que todas las sentencias PostgreSQL deben finalizar por un punto y coma (;).

Para entender mejor la clausula `SELECT` vamos a ver un ejemplo práctico.

```
SELECT * FROM employees;
```

La sentencia anterior devuelve una lista con todos los registros contenidos en la tabla `employees`:

	<code>employee_id</code> [PK] smallint	<code>last_name</code> character varying (20)	<code>first_name</code> character varying (10)
1	1	Davolio	Nancy
2	2	Fuller	Andrew
3	3	Leverling	Janet
4	4	Peacock	Margaret
5	5	Buchanan	Steven
6	6	Suyama	Michael
7	7	King	Robert
8	8	Callahan	Laura
9	9	Dodsworth	Anne

En este caso, como hemos usado el asterisco (*) en la selección de columnas, el resultado devuelto por PostgreSQL contiene todas las columnas de la tabla.

Selección de determinadas columnas

Como vimos en el ejemplo anterior al usar el asterisco (*) se seleccionan todas las columnas de la tabla. Veamos ahora un ejemplo en el que vamos a seleccionar solo unas determinadas columnas.

```
SELECT last_name, first_name, title_of_courtesy FROM employees;
```

Como vemos en la lista de columnas hemos definido qué columnas queremos recuperar, en este ejemplo son las columnas `last_name`, `first_name` y `title_of_courtesy`. El resultado que obtenemos es el siguiente:

	employee_id [PK] smallint	last_name character varying (20)	first_name character varying (10)	title character varying (30)	title_of_courtesy character varying (25)
1	1	Davolio	Nancy	Sales Representative	Ms.
2	2	Fuller	Andrew	Vice President, Sales	Dr.
3	3	Leverling	Janet	Sales Representative	Ms.
4	4	Peacock	Margaret	Sales Representative	Mrs.
5	5	Buchanan	Steven	Sales Manager	Mr.
6	6	Suyama	Michael	Sales Representative	Mr.
7	7	King	Robert	Sales Representative	Mr.
8	8	Callahan	Laura	Inside Sales Coordinator	Ms.
9	9	Dodsworth	Anne	Sales Representative	Ms.

En este caso, como hemos usado el asterisco (*) en la selección de columnas, el resultado devuelto por PostgreSQL contiene todas las columnas de la tabla.

Selección de determinadas columnas

Como vimos en el ejemplo anterior al usar el asterisco (*) se seleccionan todas las columnas de la tabla. Veamos ahora un ejemplo en el que vamos a seleccionar solo unas determinadas columnas.

```
SELECT last_name, first_name, title_of_courtesy FROM employees;
```

Como vemos en la lista de columnas hemos definido qué columnas queremos recuperar, en este ejemplo son las columnas `last_name`, `first_name` y `title_of_courtesy`. El resultado que obtenemos es el siguiente:

	last_name character varying (20)	first_name character varying (10)	title_of_courtesy character varying (25)
1	Davolio	Nancy	Ms.
2	Fuller	Andrew	Dr.
3	Leverling	Janet	Ms.
4	Peacock	Margaret	Mrs.
5	Buchanan	Steven	Mr.
6	Suyama	Michael	Mr.
7	King	Robert	Mr.
8	Callahan	Laura	Ms.
9	Dodsworth	Anne	Ms.

Selección usando expresiones

Toda la potencia de la cláusula `SELECT` reside en la potencia del lenguaje SQL de PostgreSQL. Una de estas características es el uso de expresiones para estructurar y formatear las columnas de

manera que obtengamos un conjunto de datos en la forma que nosotros deseemos. De este modo tenemos una flexibilidad total para obtener los datos y no tener que ceñirnos a la estructura que tienen en la base de datos.

Como ejemplo práctico vamos a suponer que queremos recuperar la lista de todos los empleados, pero queremos recuperar solo una columna que contenga el título de cortesía, el nombre y el apellido de cada uno:

```
SELECT title_of_courtesy || ' ' || first_name || ' ' || last_name FROM employees;
```

Observa que en la lista de columnas hemos definido una sola columna que está formada por la concatenación de los resultados de las tres columnas. La concatenación se define con doble tubo `||`. En este caso estamos concatenando el valor de dos columnas con un espacio en blanco, lo cual va a construir una cadena formateada, como puedes apreciar a continuación.

	?column? text
1	Ms. Nancy Davolio
2	Dr. Andrew Fuller
3	Ms. Janet Leverling
4	Mrs. Margaret Peacock
5	Mr. Steven Buchanan
6	Mr. Michael Suyama
7	Mr. Robert King
8	Ms. Laura Callahan
9	Ms. Anne Dodsworth

Como habíamos avanzado anteriormente, la cláusula `FROM` es opcional y puede ser omitida cuando no se consultan datos sobre una tabla. Un ejemplo es el uso en la siguiente sentencia:

```
SELECT ( 20 + 10) * 2;
```

Cuyo resultado sera:

	?column? integer
1	60

Revisión #1

Creado 15 diciembre 2023 09:31:00 por Guillermo

Actualizado 15 diciembre 2023 09:37:38 por Guillermo