

SQL DINÁMICO

Los motores de bases de datos permiten la ejecución de sentencias SQL a partir de una cadena de caracteres construida a partir de elementos variables, tales como variables locales o valores de retorno de procedimientos almacenados.

SQL Server

Ejecutamos una cadena de caracteres como una sentencia SQL a través del comando EXEC (o EXECUTE):

```
EXEC (@Nombre-de-variable)
```

En el siguiente ejemplo ejecutamos un SELECT sobre una tabla y columnas cuyo valor conocemos en tiempo de ejecución:

```
DECLARE @cadena VARCHAR(100)
DECLARE @nomTabla VARCHAR(30) = 'titles'
DECLARE @nomColumna VARCHAR(60) = 'title_id'
SET @cadena = 'SELECT ' + @nomColumna + ' FROM ' + @nomTabla;
EXEC (@cadena)
```

PostgreSQL

Ejecutamos la cadena de caracteres como una sentencia SQL a través del comando *EXECUTE Nombre-de-variable*.

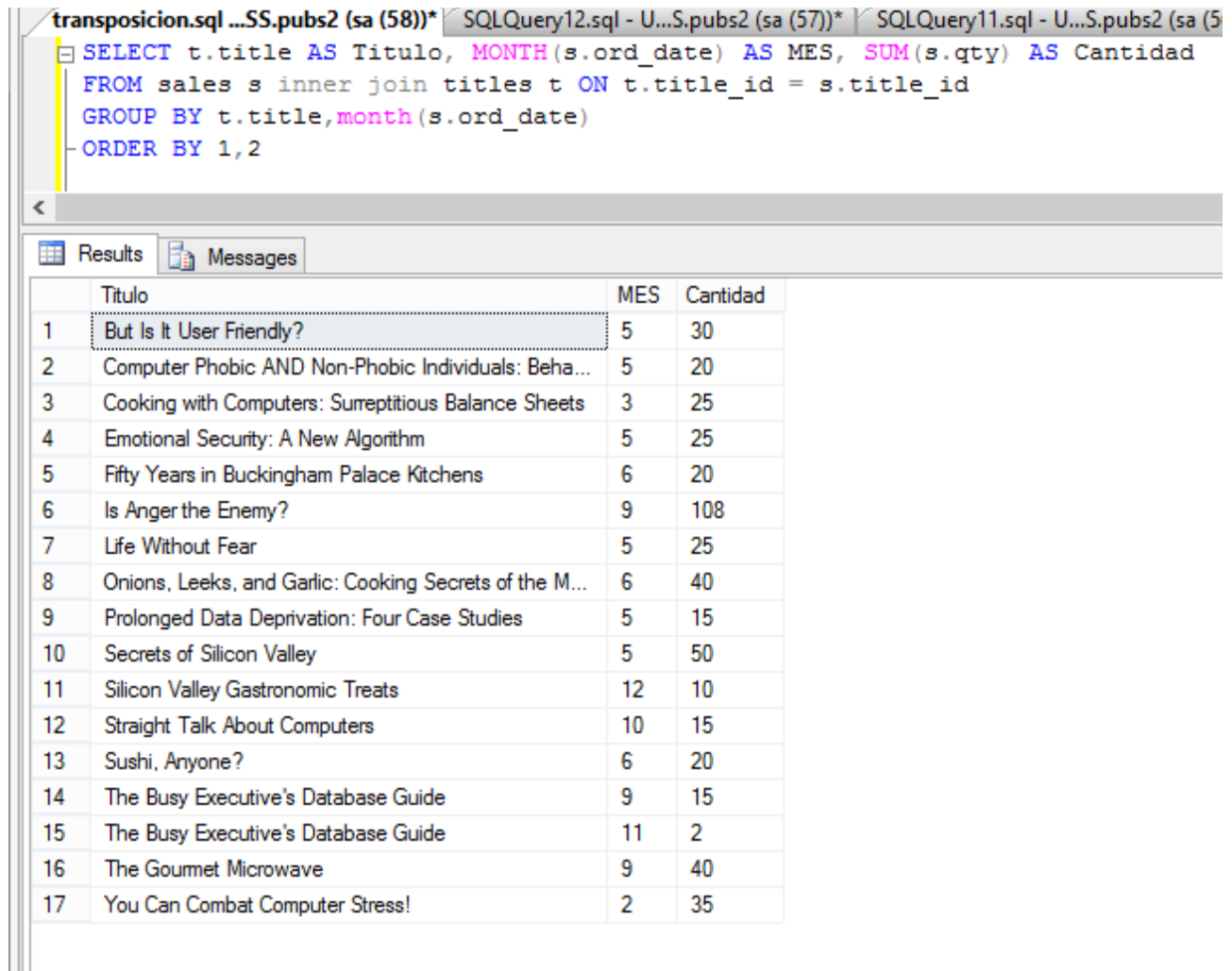
El siguiente es el mismo ejemplo de SQL dinámico en una función PL/pgSQL:

```
CREATE OR REPLACE FUNCTION sql_dinamico(varchar,varchar) RETURNS SETOF CHAR(4)
as
$BODY$
DECLARE
v_cadena VARCHAR(100);
v_nomTabla VARCHAR(30) := $1;
v_nomColumna VARCHAR(30) := $2;
BEGIN
    v_cadena := 'SELECT ' || v_nomColumna;
    v_cadena = v_cadena || ' FROM ' || v_nomTabla;
    RAISE NOTICE 'v_cadena: %', v_cadena;
    RETURN QUERY EXECUTE v_cadena;
END;
$BODY$
language plpgsql;
```

Ejemplo de transposición

Supongamos la siguiente consulta agrupando una sumatoria por un concepto y por un período. En este caso, títulos de libros vendidos por mes.

Para obtener este dato la consulta sería la siguiente:



The screenshot shows a SQL query editor with a query window titled 'transposicion.sql ...SS.pubs2 (sa (58))*'. The query is as follows:

```
SELECT t.title AS Titulo, MONTH(s.ord_date) AS MES, SUM(s.qty) AS Cantidad
FROM sales s inner join titles t ON t.title_id = s.title_id
GROUP BY t.title, month(s.ord_date)
ORDER BY 1,2
```

Below the query editor, the 'Results' tab is active, displaying a table with 17 rows of data. The table has three columns: 'Titulo', 'MES', and 'Cantidad'.

	Titulo	MES	Cantidad
1	But Is It User Friendly?	5	30
2	Computer Phobic AND Non-Phobic Individuals: Beha...	5	20
3	Cooking with Computers: Surreptitious Balance Sheets	3	25
4	Emotional Security: A New Algorithm	5	25
5	Fifty Years in Buckingham Palace Kitchens	6	20
6	Is Anger the Enemy?	9	108
7	Life Without Fear	5	25
8	Onions, Leeks, and Garlic: Cooking Secrets of the M...	6	40
9	Prolonged Data Deprivation: Four Case Studies	5	15
10	Secrets of Silicon Valley	5	50
11	Silicon Valley Gastronomic Treats	12	10
12	Straight Talk About Computers	10	15
13	Sushi, Anyone?	6	20
14	The Busy Executive's Database Guide	9	15
15	The Busy Executive's Database Guide	11	2
16	The Gourmet Microwave	9	40
17	You Can Combat Computer Stress!	2	35

Ahora bien, cómo podemos hacer para mostrar los resultados pero en columnas???

En este caso, dado que sabemos que las columnas como máximo pueden ser doce (una por mes), la siguiente consulta nos es de utilidad.

transposicion.sql ...SS.pubs2 (sa (58))* SQLQuery12.sql - U...S.pubs2 (sa (57))* SQLQuery11.sql - U...S.pubs2 (sa (56))* GRUF

```

SELECT t.title AS Titulo,
sum(CASE WHEN month(ord_date) = 1 THEN qty ELSE 0 END) AS Ene,
sum(CASE WHEN month(ord_date) = 2 THEN qty ELSE 0 END) AS Feb,
sum(CASE WHEN month(ord_date) = 3 THEN qty ELSE 0 END) AS Mar,
sum(CASE WHEN month(ord_date) = 4 THEN qty ELSE 0 END) AS Abr,
sum(CASE WHEN month(ord_date) = 5 THEN qty ELSE 0 END) AS May,
sum(CASE WHEN month(ord_date) = 6 THEN qty ELSE 0 END) AS Jun,
sum(CASE WHEN month(ord_date) = 7 THEN qty ELSE 0 END) AS Jul,
sum(CASE WHEN month(ord_date) = 8 THEN qty ELSE 0 END) AS Ago,
sum(CASE WHEN month(ord_date) = 9 THEN qty ELSE 0 END) AS Sep,
sum(CASE WHEN month(ord_date) = 10 THEN qty ELSE 0 END) AS Oct,
sum(CASE WHEN month(ord_date) = 11 THEN qty ELSE 0 END) AS Nov,
sum(CASE WHEN month(ord_date) = 12 THEN qty ELSE 0 END) AS Dic
FROM sales s inner join titles t on t.title_id = s.title_id
GROUP BY t.title
ORDER BY t.title

```

Results Messages

	Titulo	Ene	Feb	Mar	Abr	May	Jun	Jul	Ago	Sep	Oct	Nov	Dic
1	But Is It User Friendly?	0	0	0	0	30	0	0	0	0	0	0	0
2	Computer Phobic AND Non-Phobic Individuals: Beha...	0	0	0	0	20	0	0	0	0	0	0	0
3	Cooking with Computers: Surreptitious Balance Sheets	0	0	25	0	0	0	0	0	0	0	0	0
4	Emotional Security: A New Algorithm	0	0	0	0	25	0	0	0	0	0	0	0
5	Fifty Years in Buckingham Palace Kitchens	0	0	0	0	0	20	0	0	0	0	0	0
6	Is Anger the Enemy?	0	0	0	0	0	0	0	0	108	0	0	0
7	Life Without Fear	0	0	0	0	25	0	0	0	0	0	0	0
8	Onions, Leeks, and Garlic: Cooking Secrets of the M...	0	0	0	0	0	40	0	0	0	0	0	0
9	Prolonged Data Deprivation: Four Case Studies	0	0	0	0	15	0	0	0	0	0	0	0
10	Secrets of Silicon Valley	0	0	0	0	50	0	0	0	0	0	0	0
11	Silicon Valley Gastronomic Treats	0	0	0	0	0	0	0	0	0	0	0	10
12	Straight Talk About Computers	0	0	0	0	0	0	0	0	0	15	0	0

Query executed successfully.

Ahora bien, que pasa si queremos que la salida tenga columnas variables que vayan, por ejemplo desde marzo de un año hasta junio del año siguiente?

No nos quedará otra que utilizar SQL dinámico.

En el siguiente ejemplo, se crea una tabla con el total vendido por título, por cada mes, en un período de tiempo que se determina en el momento de ejecución.

Se presenta un script en dos partes, que bien podría formar parte de un procedimiento almacenado.

En la primera parte:

- Creación de tabla de trabajo
- Inserción de distintos valores de títulos para los que hay ventas en el período consultado. Esto se hace con un cursor y ejecutando, con sql dinámico, un insert por cada título.
- Bucle para generar las columnas de la salida en función del período de tiempo consultado. Se agregan las columnas necesarias, ejecutando un *alter* por cada columna (año-mes).

En la segunda parte se recorre un cursor, y se arma un *update* por cada título y por cada columna (año-mes) para las que se encuentren cantidades de ventas.

Primera parte:

```
DECLARE
    @fechaInicio datetime = '1992-03-01',
    @fechaFin datetime = '1993-06-01',
    @fecha datetime,
    @col varchar(30),
    @sqldin varchar(300),
    @sqldin_mes varchar(500),
    @tituloini varchar(80),
    @sqldin_c varchar(500);

-- Creación de tabla
CREATE TABLE titulos_por_mes
(titulo varchar(80))

-- Cursor para cargar títulos
DECLARE cur_carga_titulos CURSOR FOR
    select distinct t.title
    from sales s inner join titles t ON t.title_id = s.title_id
    where s.ord_date between @fechaInicio and @fechaFin
    order by t.title

set @sqldin = 'insert into titulos_por_mes values ('
OPEN cur_carga_titulos
FETCH NEXT FROM cur_carga_titulos INTO @tituloini
WHILE @@fetch_status = 0
BEGIN
    set @sqldin_c = @sqldin + CHAR(39) + @tituloini + CHAR(39) + ')'
    exec(@sqldin_c)
    FETCH NEXT FROM cur_carga_titulos INTO @tituloini
END
CLOSE cur_carga_titulos

-- While para agregar columnas
set @sqldin = 'alter table titulos_por_mes add '
select @fecha = @fechaInicio;
while (@fecha <= @fechaFin)
begin
    set @col = convert(varchar(4),YEAR(@fecha)) + '-' + convert(varchar(2),MONTH(@fecha));
    set @fecha = DATEADD(month,1,@fecha)
    set @sqldin_mes = ''
    set @sqldin_mes = @sqldin + ''' + @col + ''' smallint'
    exec(@sqldin_mes)
End

SELECT * FROM titulos_por_mes
```

Notas:

- *DATEADD*: función de SQLServer para sumar fechas (en este caso suma 1 mes)
- *CHAR(39)*: devuelve "'". Permite generar una comilla simple, sin problemas de compilación.

Salida primera parte:

```

        FETCH NEXT FROM cur_carga_titulos INTO @tituloIni
    -END
    CLOSE cur_carga_titulos

    set @sqldin = 'alter table titulos_por_mes add '
    select @fecha = @fechaInicio;
    while (@fecha <= @fechaFin)
    begin
        set @col = convert(varchar(4),YEAR(@fecha)) + '-' + convert(varchar(2),MONTH(@fecha));
        set @fecha = DATEADD(month,1,@fecha)
        set @sqldin_mes = ' '
        set @sqldin_mes = @sqldin + ' ' + @col + ' ' smallint'
        exec(@sqldin_mes)
    -end
    select * from titulos_por_mes

```

[illegible]

Segunda Parte:

Sobre esa tabla ya creada, se cargan los datos de cantidad en las columnas y filas según corresponda.

```
declare
    @titulo varchar(80),
    @mes varchar(30),
    @cant smallint,
    @sqldin2 varchar(300),
    @sqldin_cant varchar(500);

DECLARE cur_titulos CURSOR FOR
    select t.title, convert(varchar(4),YEAR(ord_date)) + '-' +
    convert(varchar(2),MONTH(ord_date)), SUM(qty)
    from sales s inner join titles t ON t.title_id = s.title_id
    where s.ord_date between @fechaInicio and @fechaFin
    group by t.title, convert(varchar(4),YEAR(ord_date)) + '-' +
    convert(varchar(2),MONTH(ord_date))
    order by t.title,convert(varchar(4),YEAR(ord_date)) + '-' +
    convert(varchar(2),MONTH(ord_date))

OPEN cur_titulos

SET @sqldin2 = 'update titulos_por_mes set '

FETCH NEXT FROM cur_titulos INTO @titulo,@mes,@cant
WHILE @@fetch_status = 0
BEGIN
    set @sqldin_cant =@sqldin2 + ''' + @mes + ''' + ' = ' + convert(varchar(5),@cant) +
    ' where titulo = ' + CHAR(39) + @titulo + char(39)
    exec(@sqldin_cant)
    FETCH NEXT FROM cur_titulos INTO @titulo,@mes,@cant
END
CLOSE cur_titulos
DEALLOCATE cur_titulos

SELECT * FROM titulos_por_mes
```

Salida segunda parte:

SQLQuery3.sql - UL...SS.pubs2 (sa (56)) SQLQuery2.sql - U...S.pubs2 (sa (53)) transposicion3.sql ...SS.pubs2 (sa (57))* SQLQuery1.sql - U...S.master (sa (52)) Object Explorer Details																
<pre> where s.ord_date between '1992-03-01' and '1993-06-01' group by t.title, convert(varchar(4),YEAR(ord_date)) + '-' + convert(varchar(2),MONTH(ord_date)) order by t.title,convert(varchar(4),YEAR(ord_date)) + '-' + convert(varchar(2),MONTH(ord_date)) open cur_titulos set @sqldin2 = 'update titulos_por_mes set ' FETCH NEXT FROM cur_titulos INTO @titulo,@mes,@cant WHILE @@fetch_status = 0 BEGIN set @sqldin_cant=@sqldin2 + ''' + @mes + ''' + ' = ' + convert(varchar(5),@cant) + ' where titulo = ' + CHAR(39) + @titulo + cha exec(@sqldin_cant) FETCH NEXT FROM cur_titulos INTO @titulo,@mes,@cant END CLOSE cur_titulos deallocate cur_titulos select * from titulos_por_mes </pre>																
Results	Messages															
titulo	1992-3	1992-4	1992-5	1992-6	1992-7	1992-8	1992-9	1992-10	1992-11	1992-12	1993-1	1993-2	1993-3	1993-4	1993-5	1993-6
1 But Is It User Friendly?	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	30	NULL
2 Computer Phobic AND Non-Phobic Individuals: Beha...	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	20	NULL
3 Cooking with Computers: Sureptitious Balance Sheets	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	25	NULL	NULL	NULL
4 Emotional Security: A New Algorithm	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	25	NULL
5 Fifty Years in Buckingham Palace Kitchens	NULL	NULL	NULL	20	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL
6 Life Without Fear	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	25	NULL
7 Onions, Leeks, and Garlic: Cooking Secrets of the M...	NULL	NULL	NULL	40	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL
8 Prolonged Data Deprivation: Four Case Studies	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	15	NULL
9 Secrets of Silicon Valley	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	50	NULL
10 Sushi, Anyone?	NULL	NULL	NULL	20	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL
11 You Can Combat Computer Stress!	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	35	NULL	NULL	NULL	NULL

Seguridad

Dado que lo que realmente ejecuta el motor de bases de datos es conocido en tiempo de ejecución, con su utilización nos exponemos a algunos riesgos de seguridad.

Veamos el siguiente ejemplo:

Cargamos una tabla *titles2* a partir de la tabla *titles*:

```
select * into titles2 from titles
```

Creamos el siguiente procedimiento almacenado para buscar títulos teniendo como parámetro la identificación de la editorial:

```
CREATE PROCEDURE buscaTitulos @pub VARCHAR(300)
AS
DECLARE
    @sql VARCHAR(1000)
SET @sql = 'SELECT * FROM titles WHERE pub_id = ' + @pub + ' '
select @sql
EXEC(@sql)
```

Si invocamos al procedimiento en forma correcta obtenemos el siguiente resultado:

```
exec buscaTitulos '1389'
```

	title_id	title	type	pub_id	price	advance	royalty	ytd_sales	notes
1	BU1032	The Busy Executive's Database Guide	business	1389	19.99	5000.00	10	4095	An overview of available database systems with
2	BU1111	Cooking with Computers: Surreptitious Balance Sh...	business	1389	11.95	5000.00	10	3876	Helpful hints on how to use your electronic res
3	BU7832	Straight Talk About Computers	business	1389	19.99	5000.00	10	4095	Annotated analysis of what computers can do
4	PC1035	But Is It User Friendly?	popular_comp	1389	22.95	7000.00	16	8780	A survey of software for the naive user, focusin
5	PC8888	Secrets of Silicon Valley	popular_comp	1389	20.00	8000.00	10	4095	Muckraking reporting on the world's largest co
6	PC9999	Net Etiquette	popular_comp	1389	NULL	NULL	NULL	NULL	A must-read for computer conferencing.

Pero, que sucede si lo invocamos del siguiente modo:

```
exec buscaTitulos '1389; drop table titles2'
```

La salida es idéntica:

```
exec buscaTitulos '1389; drop table titles2'
```

	title_id	title	type	pub_id	price	advance	royalty	ytd_sales	notes
1	BU1032	The Busy Executive's Database Guide	business	1389	19.99	5000.00	10	4095	An overview of available database systems with
2	BU1111	Cooking with Computers: Surreptitious Balance Sh...	business	1389	11.95	5000.00	10	3876	Helpful hints on how to use your electronic res
3	BU7832	Straight Talk About Computers	business	1389	19.99	5000.00	10	4095	Annotated analysis of what computers can do
4	PC1035	But Is It User Friendly?	popular_comp	1389	22.95	7000.00	16	8780	A survey of software for the naive user, focusin
5	PC8888	Secrets of Silicon Valley	popular_comp	1389	20.00	8000.00	10	4095	Muckraking reporting on the world's largest cor
6	PC9999	Net Etiquette	popular_comp	1389	NULL	NULL	NULL	NULL	A must-read for computer conferencing.

Pero... la tabla *titles2* no existe mas, debido a que el contenido de la variable *@sql* en este último

caso es:

```
SELECT * FROM titles WHERE pub_id = 1389; drop table titles2
```

Recomendaciones de uso:

- Utilizar SQL dinámico sólo en casos que realmente se justifiquen desde el punto de vista de programación/administración.

Ejemplos:

- patrones de búsqueda variable
 - transposición de filas a columnas sin conocimiento previo de la cantidad
 - Importación/backups/tareas de administración de bases de datos con manejo de archivos de nombre variable
- Utilizar en ambientes controlados.